TRichView Components

TRichView © trichview.com

Table of Contents

Part I TRichView reference 36

Part II Version history 38

1 New in version 24 ... 38
2 New in version 23 ... 40
3 New in version 22 ... 42
4 New in version 21 ... 43
5 New in version 20 ... 45
6 New in version 19 ... 47
7 New in version 18 ... 48
8 New in version 17 ... 51
9 New in version 16 ... 54
10 New in version 15 ... 56
11 New in version 14 ... 58
12 New in version 13 ... 62
13 New in version 12 ... 64
14 New in version 11 ... 65
15 New in version 10 ... 68
16 New in version 1.9 ... 72
17 New in version 1.8 ... 75
18 New in version 1.7 ... 77
19 New in version 1.6 ... 80
20 New in version 1.5 ... 82
21 New in version 1.4 ... 85
22 New in version 1.3 ... 86
23 New in version 1.2 ... 87

Part III Overview 89

1 Parts of documents ... 90
TRichView item types ... 90
"Checkpoints" ... 92
Items' "tags" ... 96
Hypertext ... 97
Paragraphs ... 100
Borders and background of paragraphs ... 102
Styles and styleTemplates ... 103
Hidden text ... 106
2 Basic Operations ... 106
Building TRichView document ... 107
Scrolling ... 110
Selecting ... 112

Clipboard ... 113
Searching and replacing ... 115
Obtaining Items ... 116
Modifying Items ... 117
3 Editing ... 119
Inserting at the position of caret ... 119
Undo and redo ... 120
Drag and drop ... 123
4 Graphics ... 124
Animated images ... 124
Smooth image scaling ... 125
Semitransparent objects ... 126
5 Saving and loading ... 127
Saving and loading ... 127
RVF (RichView Format) ... 129
Export to HTML ... 132
6 Internationalization ... 135
Unicode ... 135
Bidirectional text ... 137
7 Spelling check ... 137
Live spelling check ... 138
8 Technical information ... 139
Viewer vs editor ... 140
Controls - documents - items ... 141
Valid documents ... 143
Units of measurement ... 144
Custom drawing ... 146
Cursors ... 147
RVF specification ... 148
Lazarus ... 160
FireMonkey ... 161
Skia and Skia4Delphi ... 163

Part IV Overview of Items 163

1 Overview ... 164
2 Text items ... 167
3 Tabulators ... 168
4 Pictures ... 169
5 Hot-pictures ... 172
6 Breaks ... 173
7 Controls ... 174
8 Bullets ... 176
9 Hotspots ... 178
10 Tables ... 179
11 List markers ... 180
12 Labels ... 181
13 Numbered sequences ... 183
14 Endnotes ... 184
15 Footnotes ... 185

16 Sidenotes ... 186
17 Text boxes ... 188
18 References to notes ... 190
19 Page numbers ... 190
20 Page counts ... 192
21 Mathematical expressions ... 193
22 Attached files ... 195
23 Custom item types ... 195

Part V Overview of tables 195

1 Tables in TRichView ... 196
2 Addressing items in table cells ... 196
3 Table cells ... 198
4 Colors and layout of tables ... 199
5 Selection in table ... 201
6 Editing cells ... 203
7 Cell merging ... 204
8 Table operations ... 205
9 Undo/redo in tables ... 208
10 Export of tables ... 209
11 Table resizing ... 211
12 Keys in editor ... 212
13 Example: table to text ... 212
14 Example: dividing table ... 214

Part VI Components 215

1 Visual Components ... 216
TRichView ... 216
TRichView: component editor ... 224
Properties ... 226
TRichView.AllowSelection ... 228
TRichView.AnimationMode ... 228
TRichView.BackgroundBitmap ... 229
TRichView.BackgroundPicture ... 229
TRichView.BackgroundProperties ... 229
TRichView.BackgroundStyle ... 229
TRichView.BiDiMode ... 230
TRichView.BottomMargin ... 231
TRichView.ClearLeft ... 232
TRichView.ClearRight ... 232
TRichView.Color ... 233
TRichView.CPEventKind ... 233
TRichView.Cursor ... 234
TRichView.DarkMode ... 234
TRichView.Delimiters ... 235
TRichView.DocObjects ... 235
TRichView.DocParameters ... 237
TRichView.DocProperties ... 238
TRichView.Document ... 238

TRichView.DocumentHeight ... 239
TRichView.DocumentPixelsPerInch ... 239
TRichView.DoInPaletteMode ... 241
TRichView.FirstItemVisible ... 243
TRichView.FirstJumpNo ... 243
TRichView.HTMLReadProperties ... 243
TRichView.HTMLSaveProperties ... 244
TRichView.ItemCount ... 244
TRichView.LastItemVisible ... 244
TRichView.LeftMargin ... 245
TRichView.LineCount ... 245
TRichView.MarkdownProperties ... 245
TRichView.MaxLength ... 246
TRichView.MaxTextWidth ... 246
TRichView.MinTextWidth ... 246
TRichView.NoteText ... 247
TRichView.Options ... 247
TRichView.PageBreaksBeforeItems ... 250
TRichView.RightMargin ... 251
TRichView.RTFOptions ... 252
TRichView.RTFReadProperties ... 253
TRichView.RVData ... 254
TRichView.RVFOptions ... 254
TRichView.RVFParaStylesReadMode ... 258
TRichView.RVFTextStylesReadMode ... 258
TRichView.RVFWarnings ... 258
TRichView.SelectionHandlesVisible ... 260
TRichView.SingleClick ... 260
TRichView.Style ... 260
TRichView.StyleTemplateInsertMode ... 260
TRichView.TabNavigation ... 262
TRichView.TopMargin ... 262
TRichView.UseFMXThemes ... 263
TRichView.UseStyleTemplates ... 263
TRichView.UseVCLThemes ... 264
TRichView.VAlign ... 264
TRichView.VSmallStep ... 264
TRichView.WordWrap ... 265
TRichView.ZoomPercent ... 265
Methods ... 265
TRichView.AddBreak ... 269
TRichView.AddBullet ... 270
TRichView.AddCheckpoint ... 271
TRichView.AddControl ... 272
TRichView.AddFmt ... 273
TRichView.AddHotPicture ... 274
TRichView.AddHotspot ... 275
TRichView.AddItem ... 276
TRichView.AddNL -A -W ... 277
TRichView.AddPicture ... 278
TRichView.AddTab ... 280
TRichView.AddTextNL -A -W ... 281
TRichView.AppendFrom ... 282
TRichView.AppendRVFFromStream ... 283
TRichView.AssignSoftPageBreaks ... 284
TRichView.BeginOleDrag ... 284
TRichView.Clear ... 286
TRichView.ClearLiveSpellingResults ... 286

TRichView.ClearSoftPageBreaks ... 286
TRichView.ClientToDocument, DocumentToClient ... 286
TRichView.ConvertDocTo ... 287
TRichView.Copy ... 288
TRichView.CopyDef ... 288
TRichView.CopyImage ... 289
TRichView.CopyMarkdown ... 289
TRichView.CopyRTF ... 290
TRichView.CopyRVF ... 290
TRichView.CopyText -W ... 290
TRichView.Create ... 291
TRichView.DeleteItems ... 291
TRichView.DeleteMarkedStyles ... 292
TRichView.DeleteParas ... 293
TRichView.DeleteSection ... 293
TRichView.DeleteUnusedStyles ... 293
TRichView.Deselect ... 294
TRichView.Destroy ... 294
TRichView.FindCheckpointByName ... 294
TRichView.FindCheckpointByTag ... 295
TRichView.FindControlItemNo ... 295
TRichView.Format ... 295
TRichView.FormatTail ... 296
TRichView.GetBreakInfo ... 296
TRichView.GetBulletInfo ... 297
TRichView.GetCheckpointByNo ... 298
TRichView.GetCheckpointInfo ... 299
TRichView.GetCheckpointItemNo ... 299
TRichView.GetCheckpointNo ... 299
TRichView.GetCheckpointXY ... 300
TRichView.GetCheckpointY ... 300
TRichView.GetCheckpointYEx ... 301
TRichView.GetControlInfo ... 301
TRichView.GetFirstCheckpoint ... 302
TRichView.GetFocusedItem ... 302
TRichView.GetHotspotInfo ... 303
TRichView.GetItem ... 304
TRichView.GetItemAt ... 305
TRichView.GetItemCheckpoint ... 305
TRichView.GetItemCoords ... 306
TRichView.GetItemCoordsEx ... 307
TRichView.GetItemExtraIntProperty -Ex ... 307
TRichView.GetItemExtraStrProperty -Ex ... 308
TRichView.GetItemNo ... 309
TRichView.GetItemPara ... 309
TRichView.GetItemStyle ... 310
TRichView.GetItemTag ... 310
TRichView.GetItemText -A -W ... 310
TRichView.GetItemVAlign ... 311
TRichView.GetJumpPointItemNo ... 312
TRichView.GetJumpPointLocation ... 312
TRichView.GetJumpPointY ... 313
TRichView.GetLastCheckpoint ... 313
TRichView.GetLineNo ... 314
TRichView.GetListMarkerInfo ... 315
TRichView.GetNextCheckpoint ... 315
TRichView.GetOffsAfterItem ... 316
TRichView.GetOffsBeforeItem ... 317

TRichView.GetPictureInfo ... 318
TRichView.GetPrevCheckpoint ... 319
TRichView.GetRealDocumentPixelsPerInch ... 319
TRichView.GetSelectedImage ... 319
TRichView.GetSelectionBounds ... 320
TRichView.GetSelText -A -W ... 321
TRichView.GetTextInfo ... 321
TRichView.GetWordAt -A -W ... 322
TRichView.InsertRVFFromStream ... 323
TRichView.IsFromNewLine ... 325
TRichView.IsParaStart ... 325
TRichView.LiveSpellingValidateWord ... 326
TRichView.LoadDocX ... 326
TRichView.LoadDocXFromStream ... 327
TRichView.LoadFromFile -Ex ... 328
TRichView.LoadFromStream -Ex ... 329
TRichView.LoadHTML ... 330
TRichView.LoadHTMLFromStream ... 332
TRichView.LoadMarkdown ... 332
TRichView.LoadMarkdownFromStream ... 333
TRichView.LoadRTF ... 334
TRichView.LoadRTFFromStream ... 335
TRichView.LoadRVF ... 336
TRichView.LoadRVFFromStream ... 337
TRichView.LoadText -W ... 338
TRichView.LoadTextFromStream -W ... 339
TRichView.MarkStylesInUse ... 339
TRichView.Reformat ... 339
TRichView.RefreshListMarkers ... 340
TRichView.RefreshListSequences ... 340
TRichView.RemoveCheckpoint ... 340
TRichView.ResetAnimation ... 341
TRichView.SaveDocX ... 341
TRichView.SaveDocXToStream ... 342
TRichView.SaveHTML ... 342
TRichView.SaveHTML (deprecated) ... 343
TRichView.SaveHTMLEx (deprecated) ... 345
TRichView.SaveHTMLToStream ... 346
TRichView.SaveHTMLToStream (deprecated) ... 347
TRichView.SaveHTMLToStreamEx (deprecated) ... 348
TRichView.SaveMarkdown ... 348
TRichView.SaveMarkdownToStream ... 349
TRichView.SavePicture ... 349
TRichView.SaveRTF ... 350
TRichView.SaveRTFToStream ... 351
TRichView.SaveRVF ... 351
TRichView.SaveRVFToStream ... 352
TRichView.SaveText -W ... 352
TRichView.SaveTextToStream -W ... 353
TRichView.SearchText -A -W ... 354
TRichView.SelectAll ... 355
TRichView.SelectControl ... 355
TRichView.SelectionExists ... 356
TRichView.SelectWordAt ... 356
TRichView.SetAddParagraphMode ... 357
TRichView.SetBreakInfo ... 358
TRichView.SetBulletInfo ... 359
TRichView.SetCheckpointInfo ... 360

TRichView.SetControlInfo ... 361
TRichView.SetFooter ... 362
TRichView.SetHeader ... 363
TRichView.SetHotspotInfo ... 364
TRichView.SetItemExtraIntProperty -Ex ... 366
TRichView.SetItemExtraStrProperty -Ex ... 367
TRichView.SetItemTag ... 367
TRichView.SetItemText -A -W ... 368
TRichView.SetItemVAlign ... 369
TRichView.SetListMarkerInfo ... 369
TRichView.SetPictureInfo ... 370
TRichView.SetSelectionBounds ... 372
TRichView.StartAnimation ... 374
TRichView.StartLiveSpelling ... 374
TRichView.StopAnimation ... 375
TRichView.StoreSearchResult ... 375
TRichView.UpdatePaletteInfo ... 375
Events ... 376
TRichView.OnAddStyle ... 377
TRichView.OnAfterDrawImage ... 378
TRichView.OnAssignImageFileName ... 379
TRichView.OnCheckpointVisible ... 379
TRichView.OnCMHintShow ... 380
TRichView.OnControlAction ... 380
TRichView.OnCopy ... 381
TRichView.OnGetItemCursor ... 382
TRichView.OnGetSpellingSuggestions ... 382
TRichView.OnHTMLSaveImage ... 383
TRichView.OnImportFile ... 385
TRichView.OnImportPicture ... 386
TRichView.OnItemAction ... 387
TRichView.OnItemHint ... 388
TRichView.OnJump ... 390
TRichView.OnLoadCustomFormat ... 390
TRichView.OnLoadDocument ... 391
TRichView.OnNewDocument ... 391
TRichView.OnPaint ... 392
TRichView.OnProgress ... 392
TRichView.OnReadField ... 393
TRichView.OnReadHyperlink ... 395
TRichView.OnReadMergeField ... 398
TRichView.OnRVDblClick ... 399
TRichView.OnRVFControlNeeded ... 400
TRichView.OnRVFImageListNeeded ... 400
TRichView.OnRVFPictureNeeded ... 401
TRichView.OnRVMouseDown ... 402
TRichView.OnRVMouseMove ... 403
TRichView.OnRVMouseUp ... 403
TRichView.OnRVRightClick ... 404
TRichView.OnSaveComponentToFile ... 405
TRichView.OnSaveDocXExtra ... 407
TRichView.OnSaveHTMLExtra ... 409
TRichView.OnSaveImage2 ... 410
TRichView.OnSaveItemToFile ... 411
TRichView.OnSaveParaToHTML ... 413
TRichView.OnSaveRTFExtra ... 414
TRichView.OnSelect ... 416
TRichView.OnSpellingCheck ... 416

TRichView.OnStyleTemplatesChange 417
TRichView.OnTextFound 418
TRichView.OnWriteHyperlink 418
TRichView.OnWriteObjectProperties 420
Classes of properties 422
TRVDocParameters 422
Properties 423
TRVDocParameters.Author 424
TRVDocParameters.BottomMargin 424
TRVDocParameters.Comments 424
TRVDocParameters.FacingPages 424
TRVDocParameters.FooterY 425
TRVDocParameters.HeaderY 425
TRVDocParameters.LeftMargin 425
TRVDocParameters.MirrorMargins 426
TRVDocParameters.Orientation 426
TRVDocParameters.PageHeight 426
TRVDocParameters.PageWidth 426
TRVDocParameters.RightMargin 426
TRVDocParameters.Title 427
TRVDocParameters.TitlePage 427
TRVDocParameters.TopMargin 427
TRVDocParameters.Units 428
TRVDocParameters.ZoomMode 428
TRVDocParameters.ZoomPercent 428
Methods 428
TRVDocParameters.ConvertToUnits 428
TRVDocParameters.Reset 429
TRVDocParameters.ResetLayout 429
TRVDocParameters.UnitsPerInch 429
TRVDocumentProperty 429
Properties 429
TRVDocumentProperty.AllowMarkdown 430
TRVDocumentProperty.AutoDeleteUnusedStyles 430
TRVDocumentProperty.CodePage 431
TRVDocumentProperty.FieldFormat 431
TRVDocumentProperty.IgnoreEscape 432
TRVHTMLReaderProperties 432
Properties 432
TRVHTMLReaderProperties.BasePathLinks 433
TRVHTMLReaderProperties.EnforceListLevels 433
TRVHTMLReaderProperties.FontSizePrecision 434
TRVHTMLReaderProperties.IDAsCheckpoints 434
TRVHTMLReaderProperties.ReadBackground 434
TRVHTMLReaderProperties.ReadDocParameters 435
TRVHTMLReaderProperties.SkipHiddenText 435
TRVHTMLReaderProperties.XHTMLCheck 436
TRVHTMLSaveProperties 436
Properties 437
TRVHTMLSaveProperties.CheckpointsPrefix 437
TRVHTMLSaveProperties.CSSOptions 438
TRVHTMLSaveProperties.CSSSavingType 438
TRVHTMLSaveProperties.Default0Style 439
TRVHTMLSaveProperties.Encoding 439
TRVHTMLSaveProperties.EncodingStr 439
TRVHTMLSaveProperties.ExternalCSSFileName 440
TRVHTMLSaveProperties.ExtraStyles 440
TRVHTMLSaveProperties.HTMLSavingType 441

TRVHTMLSaveProperties.HTMLVersion ... 441
TRVHTMLSaveProperties.ImageOptions ... 442
TRVHTMLSaveProperties.ImagesPrefix ... 443
TRVHTMLSaveProperties.ListMarkerSavingType ... 444
TRVHTMLSaveProperties.SaveHeaderAndFooter ... 445
TRVHTMLSaveProperties.UseCheckpointNames ... 445
TRVMarkdownDefaultItemProperties ... 445
Properties ... 446
TRVMarkdownDefaultItemProperties.BlockQuoteBackColor ... 446
TRVMarkdownDefaultItemProperties.BreakColor ... 446
TRVMarkdownDefaultItemProperties.CodeBlockBackColor ... 446
TRVMarkdownDefaultItemProperties.ImageVAlign ... 447
TRVMarkdownDefaultItemProperties.ListMarkerIndentPix, ListLeftIndentPix ... 447
TRVMarkdownDefaultItemProperties.TableCellWidthPix ... 447
TRVMarkdownProperties ... 447
Properties ... 451
TRVMarkdownProperties.DefaultItemProperties ... 452
TRVMarkdownProperties.Extensions ... 452
TRVMarkdownProperties.ImagesPrefix ... 453
TRVMarkdownProperties.LineWidth ... 453
TRVMarkdownProperties.LoadOptions ... 453
TRVMarkdownProperties.NotesSavedAsFootnotes ... 455
TRVMarkdownProperties.SaveHiddenText ... 456
TRVMarkdownProperties.SaveOptions ... 456
TRVRTFReaderProperties ... 457
Properties ... 458
TRVRTFReaderProperties.AutoHideTableGridLines ... 459
TRVRTFReaderProperties.BasePathLinks ... 459
TRVRTFReaderProperties.CanReuseNumberedLists ... 459
TRVRTFReaderProperties.CharsetForUnicode ... 460
TRVRTFReaderProperties.ConvertHighlight ... 460
TRVRTFReaderProperties.ConvertSymbolFonts ... 461
TRVRTFReaderProperties.DefCharsetCodePage ... 461
TRVRTFReaderProperties.DocXErrorCode ... 462
TRVRTFReaderProperties.ExtractMetafileBitmaps ... 462
TRVRTFReaderProperties.IgnoreBookmarks ... 463
TRVRTFReaderProperties.IgnoreFields ... 463
TRVRTFReaderProperties.IgnoreNotes ... 463
TRVRTFReaderProperties.IgnorePictures ... 463
TRVRTFReaderProperties.IgnoreSequences ... 463
TRVRTFReaderProperties.IgnoreTables ... 464
TRVRTFReaderProperties.LineBreaksAsParagraphs ... 464
TRVRTFReaderProperties.ParaStyleMode ... 464
TRVRTFReaderProperties.ParaStyleNo ... 465
TRVRTFReaderProperties.ReadDocParameters ... 465
TRVRTFReaderProperties.ReadNoteNumberingType ... 465
TRVRTFReaderProperties.RTFErrorCode ... 466
TRVRTFReaderProperties.SkipHiddenText ... 466
TRVRTFReaderProperties.TextStyleMode ... 466
TRVRTFReaderProperties.TextStyleNo ... 467
TRVRTFReaderProperties.UnicodeMode ... 467
TRVRTFReaderProperties.UseCharsetForUnicode ... 468
TRVRTFReaderProperties.UseHypertextStyles ... 468
TRVRTFReaderProperties.UseSingleCellPadding ... 468
Examples ... 468
Loading UTF-8 files ... 468
TRichViewEdit ... 469
Properties ... 475

TRichViewEdit.AcceptDragDropFormats, AcceptPasteFormats 478
TRichViewEdit.ActualCurTextStyleNo 480
TRichViewEdit.CheckSpelling 480
TRichViewEdit.CurItemNo 480
TRichViewEdit.CurItemStyle 481
TRichViewEdit.CurParaStyleNo 481
TRichViewEdit.CurTextStyleNo 482
TRichViewEdit.CustomCaretInterval 483
TRichViewEdit.DefaultPictureVAlign 483
TRichViewEdit.EditorOptions 483
TRichViewEdit.ForceFieldHighlight 486
TRichViewEdit.KeyboardType 487
TRichViewEdit.LiveSpellingMode 487
TRichViewEdit.Modified 487
TRichViewEdit.OffsetInCurItem 488
TRichViewEdit.ReadOnly 488
TRichViewEdit.SmartPopupProperties 489
TRichViewEdit.SmartPopupVisible 489
TRichViewEdit.TopLevelEditor 489
TRichViewEdit.UndoLimit 490
Methods 490
TRichViewEdit.AddSpellingMenuItems 497
TRichViewEdit.AdjustControlPlacement 497
TRichViewEdit.AdjustControlPlacement2 497
TRichViewEdit.ApplyListStyle 498
TRichViewEdit.ApplyParaStyle 498
TRichViewEdit.ApplyParaStyleConversion 499
TRichViewEdit.ApplyParaStyleTemplate 500
TRichViewEdit.ApplyStyleConversion 501
TRichViewEdit.ApplyStyleTemplate 501
TRichViewEdit.ApplyTextStyle 502
TRichViewEdit.ApplyTextStyleTemplate 502
TRichViewEdit.BeginItemModify 503
TRichViewEdit.BeginUndoCustomGroup 504
TRichViewEdit.BeginUndoCustomGroup2, EndUndoCustomGroup2 504
TRichViewEdit.BeginUndoGroup 505
TRichViewEdit.BeginUndoGroup2, EndUndoGroup2 505
TRichViewEdit.BeginUpdate 506
TRichViewEdit.CanChange 506
TRichViewEdit.CanPaste 507
TRichViewEdit.CanPasteHTML 507
TRichViewEdit.CanPasteRTF 507
TRichViewEdit.CanPasteRVF 507
TRichViewEdit.Change 508
TRichViewEdit.ChangeListLevels 508
TRichViewEdit.ChangeStyleTemplates 508
TRichViewEdit.Clear 509
TRichViewEdit.ClearTextFlow 509
TRichViewEdit.ClearUndo 509
TRichViewEdit.ConvertToHotPicture 510
TRichViewEdit.ConvertToPicture 510
TRichViewEdit.Create 510
TRichViewEdit.CutDef 511
TRichViewEdit.DeleteSelection 511
TRichViewEdit.Destroy 511
TRichViewEdit.EndItemModify 511
TRichViewEdit.EndUpdate 512
TRichViewEdit.GetCheckpointAtCaret 512

TRichViewEdit.GetCurrentBreakInfo ... 512
TRichViewEdit.GetCurrentBulletInfo ... 513
TRichViewEdit.GetCurrentCheckpoint ... 514
TRichViewEdit.GetCurrentControlInfo ... 515
TRichViewEdit.GetCurrentHotspotInfo ... 515
TRichViewEdit.GetCurrentItem ... 517
TRichViewEdit.GetCurrentItemEx ... 517
TRichViewEdit.GetCurrentItemExtraIntProperty -Ex ... 518
TRichViewEdit.GetCurrentItemExtraStrProperty -Ex ... 519
TRichViewEdit.GetCurrentItemText -A -W ... 519
TRichViewEdit.GetCurrentItemVAlign ... 520
TRichViewEdit.GetCurrentLineCol ... 521
TRichViewEdit.GetCurrentMisspelling ... 521
TRichViewEdit.GetCurrentPictureInfo ... 522
TRichViewEdit.GetCurrentTag ... 522
TRichViewEdit.GetCurrentTextInfo ... 523
TRichViewEdit.InsertBreak ... 523
TRichViewEdit.InsertBullet ... 524
TRichViewEdit.InsertCheckpoint ... 525
TRichViewEdit.InsertControl ... 526
TRichViewEdit.InsertDocXFromFileEd ... 526
TRichViewEdit.InsertDocXFromStreamEd ... 527
TRichViewEdit.InsertHotPicture ... 528
TRichViewEdit.InsertHotspot ... 528
TRichViewEdit.InsertHTMLFromFileEd ... 529
TRichViewEdit.InsertHTMLFromStreamEd ... 530
TRichViewEdit.InsertItem ... 531
TRichViewEdit.InsertMarkdownFromFileEd ... 532
TRichViewEdit.InsertMarkdownFromStreamEd ... 532
TRichViewEdit.InsertOEMTextFromFile ... 533
TRichViewEdit.InsertPageBreak ... 534
TRichViewEdit.InsertPicture ... 534
TRichViewEdit.InsertRTFFromFileEd ... 536
TRichViewEdit.InsertRTFFromStreamEd ... 536
TRichViewEdit.InsertRVFFromFileEd ... 537
TRichViewEdit.InsertRVFFromStreamEd ... 538
TRichViewEdit.InsertStringTag -A -W ... 539
TRichViewEdit.InsertTab ... 540
TRichViewEdit.InsertText -A -W ... 540
TRichViewEdit.InsertTextFromFile -W ... 541
TRichViewEdit.MoveCaret ... 542
TRichViewEdit.Paste ... 543
TRichViewEdit.PasteBitmap ... 544
TRichViewEdit.PasteGraphicFiles ... 544
TRichViewEdit.PasteHTML ... 545
TRichViewEdit.PasteMarkdown ... 546
TRichViewEdit.PasteMetafile ... 546
TRichViewEdit.PasteRTF ... 547
TRichViewEdit.PasteRVF ... 547
TRichViewEdit.PasteText -A -W ... 548
TRichViewEdit.PasteURL ... 549
TRichViewEdit.Redo ... 549
TRichViewEdit.RedoAction ... 549
TRichViewEdit.RedoName ... 550
TRichViewEdit.RemoveCheckpointAtCaret ... 550
TRichViewEdit.RemoveCheckpointEd ... 550
TRichViewEdit.RemoveCurrentCheckpoint ... 551
TRichViewEdit.RemoveCurrentPageBreak ... 551

TRichViewEdit.RemoveLists ... 552
TRichViewEdit.ResizeControl ... 552
TRichViewEdit.ResizeCurrentControl ... 552
TRichViewEdit.SearchText -W ... 553
TRichViewEdit.SelectCurrentLine ... 554
TRichViewEdit.SelectCurrentWord ... 555
TRichViewEdit.Set*PropertyEd ... 555
TRichViewEdit.SetBackgroundImageEd ... 556
TRichViewEdit.SetBackgroundPropertyEd ... 557
TRichViewEdit.SetBreakInfoEd ... 557
TRichViewEdit.SetBulletInfoEd ... 558
TRichViewEdit.SetCheckpointInfoEd ... 560
TRichViewEdit.SetControlInfoEd ... 560
TRichViewEdit.SetCurrentBreakInfo ... 561
TRichViewEdit.SetCurrentBulletInfo ... 562
TRichViewEdit.SetCurrentCheckpointInfo ... 563
TRichViewEdit.SetCurrentControlInfo ... 564
TRichViewEdit.SetCurrentHotspotInfo ... 565
TRichViewEdit.SetCurrentItemExtraIntProperty -Ex ... 566
TRichViewEdit.SetCurrentItemExtraStrProperty -Ex ... 567
TRichViewEdit.SetCurrentItemText -A -W ... 568
TRichViewEdit.SetCurrentItemVAlign ... 569
TRichViewEdit.SetCurrentPictureInfo ... 569
TRichViewEdit.SetCurrentTag ... 570
TRichViewEdit.SetHotspotInfoEd ... 570
TRichViewEdit.SetItemExtraIntProperty-Ed -ExEd ... 572
TRichViewEdit.SetItemExtraStrProperty-Ed -ExEd ... 573
TRichViewEdit.SetItemTagEd ... 574
TRichViewEdit.SetItemTextEd -A -W ... 574
TRichViewEdit.SetItemVAlignEd ... 575
TRichViewEdit.SetPictureInfoEd ... 576
TRichViewEdit.SetUndoGroupMode ... 577
TRichViewEdit.Undo ... 577
TRichViewEdit.UndoAction ... 577
TRichViewEdit.UndoName ... 578
Events ... 578
TRichViewEdit.OnBeforeOleDrop, OnAfterOleDrop ... 580
TRichViewEdit.OnCaretGetOut ... 581
TRichViewEdit.OnCaretMove ... 582
TRichViewEdit.OnChange ... 582
TRichViewEdit.OnChanging ... 582
TRichViewEdit.OnCheckStickingItems ... 583
TRichViewEdit.OnCurParaStyleChanged ... 583
TRichViewEdit.OnCurTextStyleChanged ... 584
TRichViewEdit.OnDrawCurrentLine ... 584
TRichViewEdit.OnDrawCustomCaret ... 585
TRichViewEdit.OnDropFile ... 586
TRichViewEdit.OnDropFiles ... 586
TRichViewEdit.OnItemResize ... 587
TRichViewEdit.OnItemTextEdit ... 588
TRichViewEdit.OnMeasureCustomCaret ... 588
TRichViewEdit.OnOleDragEnter ... 589
TRichViewEdit.OnOleDragLeave ... 589
TRichViewEdit.OnOleDragOver ... 590
TRichViewEdit.OnOleDrop ... 591
TRichViewEdit.OnParaStyleConversion ... 592
TRichViewEdit.OnPaste ... 594
TRichViewEdit.OnSaveCustomFormat ... 595

TRichViewEdit.OnSmartPopupClick 595
TRichViewEdit.OnStyleConversion 595
Classes of properties 598
TRVSmartPopupProperties 598
Properties 598
TRVSmartPopupProperties.ButtonType 599
TRVSmartPopupProperties.Color 599
TRVSmartPopupProperties.Hint 600
TRVSmartPopupProperties.HoverColor 600
TRVSmartPopupProperties.HoverLineColor 600
TRVSmartPopupProperties.ImageIndex 600
TRVSmartPopupProperties.ImageList 600
TRVSmartPopupProperties.LineColor 601
TRVSmartPopupProperties.Menu 601
TRVSmartPopupProperties.Popup 601
TRVSmartPopupProperties.Position 601
TRVSmartPopupProperties.ShortCut 601
Methods 602
TRVSmartPopupProperties.SetButtonState 602
Examples 602
Moving caret to the beginning of paragraph 602
OnDropFile Example 603
TDBRichView 604
Properties 606
TDBRichView.AllowMarkdown 608
TDBRichView.AutoDeleteUnusedStyles 608
TDBRichView.AutoDisplay 609
TDBRichView.CodePage 609
TDBRichView.DataField 610
TDBRichView.DataSource 610
TDBRichView.Field 610
Methods 610
TDBRichView.Create 614
TDBRichView.Destroy 614
TDBRichView.LoadField 614
Events 614
TDBRichViewEdit 616
Properties 618
TDBRichViewEdit.AllowMarkdown 621
TDBRichViewEdit.AutoDeleteUnusedStyles 621
TDBRichViewEdit.AutoDisplay 622
TDBRichViewEdit.CodePage 622
TDBRichViewEdit.DataField 622
TDBRichViewEdit.DataSource 623
TDBRichViewEdit.Field 623
TDBRichViewEdit.FieldFormat 623
TDBRichViewEdit.IgnoreEscape 624
TDBRichViewEdit.ReadOnly 624
Methods 624
TDBRichViewEdit.Change 631
TDBRichViewEdit.Create 631
TDBRichViewEdit.Destroy 631
TDBRichViewEdit.LoadField 631
Events 632
Examples 634
Using viewer-style methods in DBRichViewEdit 634
Inserting controls that can modify themselves in DBRichViewEdit 634
TRVPrintPreview 635

Properties ... 636
TRVPrintPreview.RVPrint ... 638
TRVPrintPreview.CachePageImage ... 638
Methods ... 638
Events ... 638
2 Additional Components ... 639
TRVChat ... 640
Properties ... 641
TRVChat.Callout ... 641
TRVChat.ChatMaxWidth ... 641
TRVChat.ChatRichView ... 642
TRVChat.DateLine ... 642
TRVChat.FileIcons ... 642
TRVChat.Fonts ... 642
TRVChat.HighlightedMessage ... 642
TRVChat.LinkColor ... 643
TRVChat.Messages ... 643
TRVChat.MessageStatusLine ... 643
TRVChat.Quote ... 643
TRVChat.Texts ... 643
TRVChat.UserPicture ... 644
TRVChat.Users ... 644
Methods ... 644
TRVChat.AddMessage ... 645
TRVChat.AddStyles ... 646
TRVChat.ChangeMessageImage ... 646
TRVChat.ChangeMessageText ... 647
TRVChat.ClearMessages ... 647
TRVChat.CopyToClipboard ... 647
TRVChat.DeleteMessage ... 647
TRVChat.FindMessages ... 648
TRVChat.GetAutoQuote, GetAutoQuoteFromItemNo ... 648
TRVChat.GetFirstVisibleMessage ... 649
TRVChat.GetImageAtCursor ... 649
TRVChat.GetMessage, GetMessageAndImage ... 649
TRVChat.GetMessageAt* ... 649
TRVChat.GetMessageImage, GetMessageImageEx ... 650
TRVChat.GetMessageItemNo, GetMessageTextItemNo ... 650
TRVChat.GetMessagesAsText ... 651
TRVChat.GetSelectedMessage, GetRangeOfSelectedMessages ... 651
TRVChat.GetSelectedTextForQuote, HasSelectedTextForQuote ... 652
TRVChat.InsertMessage ... 652
TRVChat.MarkMessageAsDeleted ... 653
TRVChat.MoveMessage ... 653
TRVChat.ScrollToMessage ... 654
TRVChat.UpdateTextStyles, UpdateParaStyles ... 654
Events ... 654
TRVChat.OnClickDate ... 654
TRVChat.OnClickFile ... 654
TRVChat.OnClickImage ... 655
TRVChat.OnClickLink ... 655
TRVChat.OnClickUser ... 655
TRVChat.OnHighlightedMessageChange ... 655
Related classes and records ... 656
TRVChatCalloutProperties ... 657
Properties ... 657
TRVChatCalloutProperties.ArrowHeight, ArrowWidth ... 657

TRVChatCalloutProperties.CalloutType 658
TRVChatCalloutProperties.CornerRadiusLarge, CornerRadiusSmall 658
TRVChatCalloutProperties.Margin 658
TRVChatCalloutProperties.MaxWidth, MinWidth 658
TRVChatCalloutProperties.MinOppositeMargin 659
TRVChatCalloutProperties.Padding 659
TRVChatCalloutProperties.ShadowOpacity 659
TRVChatCalloutProperties.Spacing 659
TRVChatDateProperties 660
Properties 660
TRVChatDateProperties.BackColor 660
TRVChatDateProperties.DateFormatNew, DateFormatOld 660
TRVChatDateProperties.MarginAbove, MarginBelow 661
TRVChatDateProperties.PaddingHorizontal, PaddingVertical 661
TRVChatFileCollection 661
Properties 661
TRVChatFileCollection.Items 662
TRVChatFileIconsProperties 662
Properties 662
TRVChatFileIconsProperties.IconColor, IconSize, IconPadding 662
TRVChatFileIconsProperties.PictureArchive, etc. 663
TRVChatFileIconsProperties.ScalePictures 663
TRVChatFileItem 663
Properties 664
TRVChatFileItem.DisplayName 664
TRVChatFileItem.FileName 664
TRVChatFileItem.FileSize 664
TRVChatFileItem.FileType 664
TRVChatFonts 664
TRVChatMessage 665
Properties 666
TRVChatMessage.Id 666
TRVChatMessage.DateTime 666
TRVChatMessage.EditCount 666
TRVChatMessage.GlobalIndex 667
TRVChatMessage.Text 667
TRVChatMessage.State 667
TRVChatMessage.User 667
TRVChatMessage.Quote 667
TRVChatMessage.Files 668
TRVChatMessageCollection 668
Properties 668
TRVChatMessageCollection.Items 668
Methods 668
TRVChatMessageCollection.FindByDate, FindItemByDate 669
TRVChatMessageCollection.FindByGlobalIndex, FindItemByGlobalIndex 669
TRVChatMessageCollection.FindIById, FindItemById 669
TRVChatMessageCollection.FindInsertionPlace 670
TRVChatMessageExtra 670
Properties 670
TRVChatMessageExtra.AutoDestroy 670
TRVChatMessageExtraFiles 670
Properties 671
TRVChatMessageExtraFiles.Files 671
Methods 671
TRVChatMessageExtraFiles.AddFile 671
TRVChatMessageExtraFiles.AssignDefaultFileTypes 672
TRVChatMessageExtraFiles.Create 672

TRVChatMessageExtraImage ... 672
Properties ... 672
TRVChatMessageExtraImage.Image ... 673
TRVChatMessageExtraImage.ImageText ... 673
TRVChatMessageExtraImage.MustAddCopyOfImage ... 673
Methods ... 673
TRVChatMessageExtraImage.Create ... 673
TRVChatMessageExtraQuote ... 673
Properties ... 674
TRVChatMessageExtraQuote.Msg ... 674
TRVChatMessageExtraQuote.Text ... 674
TRVChatMessageExtraQuote.User ... 674
Methods ... 674
TRVChatMessageExtraQuote.Create ... 674
TRVChatMessageInsertionInfo ... 675
TRVChatMessageQuote ... 675
Properties ... 675
TRVChatMessageQuote.Msg ... 676
TRVChatMessageQuote.QuoteType ... 676
TRVChatMessageQuote.Text ... 676
TRVChatMessageQuote.User ... 676
TRVChatMessageStatusLine ... 676
Properties ... 677
TRVChatMessageStatusLine.Images ... 677
TRVChatMessageStatusLine.ImagesHeight, ImagesWidth ... 677
TRVChatMessageStatusLine.MinStatusLineHeight ... 677
TRVChatMessageStatusLine.StatusLineContent ... 678
TRVChatMessageStatusLine.StatusLineLocation ... 678
TRVChatMessageStatusLine's ImageIndexes ... 679
TRVChatQuoteProperties ... 679
Properties ... 679
TRVChatQuoteProperties.CompactQuotes ... 679
TRVChatQuoteProperties.CornerRadius ... 680
TRVChatQuoteProperties.MaxAutoQuoteLength ... 680
TRVChatQuoteProperties.Padding ... 680
TRVChatUITexts ... 680
TRVChatUser ... 681
Properties ... 681
TRVChatUser.FillColor ... 682
TRVChatUser.Id ... 682
TRVChatUser.IsRightSide ... 682
TRVChatUser.MessageInfoColor ... 682
TRVChatUser.Name ... 682
TRVChatUser.NameColor ... 682
TRVChatUser.Picture ... 683
TRVChatUser.QuoteFillColor ... 683
TRVChatUser.ShowNameInMessages ... 683
TRVChatUser.ShowPicture ... 683
TRVChatUserCollection ... 683
Properties ... 684
TRVChatUserCollection.Items ... 684
Methods ... 684
TRVChatUserCollection.FindIById, FindItemById ... 684
TRVChatUserPictureProperties ... 684
Properties ... 685
TRVChatUserPictureProperties.Position ... 685
TRVChatUserPictureProperties.Size ... 685
TRVDataSourceLink ... 685

Properties ... 686
TRVDataSourceLink.DataSource ... 686
TRVDataSourceLink.Editor ... 686
Methods ... 687
TRVDataSourceLink.Execute ... 687
TRVOfficeConverter ... 687
Properties ... 688
TRVOfficeConverter.ErrorCode ... 689
TRVOfficeConverter.ExcludeDocX-Converter ... 690
TRVOfficeConverter.ExcludeHTML-Converter ... 690
TRVOfficeConverter.ExportConverters ... 690
TRVOfficeConverter.ExtensionsInFilter ... 690
TRVOfficeConverter.ImportConverters ... 691
TRVOfficeConverter.PreviewMode ... 691
TRVOfficeConverter.Stream ... 691
Methods ... 691
TRVOfficeConverter.Create ... 692
TRVOfficeConverter.Destroy ... 692
TRVOfficeConverter.ExportRTF ... 692
TRVOfficeConverter.ExportRV ... 693
TRVOfficeConverter.GetExportFilter ... 693
TRVOfficeConverter.GetImportFilter ... 693
TRVOfficeConverter.ImportRTF ... 694
TRVOfficeConverter.ImportRV ... 694
TRVOfficeConverter.IsValidImporter ... 695
Events ... 695
TRVOfficeConverter.OnConverting ... 695
Classes of properties ... 696
TRVOfficeCnvList ... 696
TRVOfficeConverterInfo ... 696
TRVPrint ... 696
Properties ... 702
TRVPrint.BestDPI ... 702
TRVPrint.ClipMargins ... 703
TRVPrint.FacingPages ... 703
TRVPrint.FixMarginsMode ... 703
TRVPrint.FooterY ... 704
TRVPrint.HeaderY ... 705
TRVPrint.Margins ... 705
TRVPrint.MirrorMargins ... 706
TRVPrint.Preview100PercentHeight ... 706
TRVPrint.Preview100PercentWidth ... 706
TRVPrint.TitlePage ... 706
TRVPrint.Units ... 707
TRVPrint.VirtualPrinter ... 707
Methods ... 708
TRVPrint.AssignDocParameters ... 708
TRVPrint.AssignSource ... 709
TRVPrint.ContinuousPrint ... 709
TRVPrint.ConvertToUnits ... 710
TRVPrint.Create ... 710
TRVPrint.DrawPage, DrawPageAt ... 710
TRVPrint.DrawPreview ... 711
TRVPrint.FormatPages ... 711
TRVPrint.GetFooterRect ... 712
TRVPrint.GetHeaderRect ... 712
TRVPrint.MakePreview ... 712
TRVPrint.MakeScaledPreview ... 713

TRVPrint.Print ... 713
TRVPrint.PrintPages ... 714
TRVPrint.SavePDF ... 715
TRVPrint.SetFooter ... 715
TRVPrint.SetHeader ... 716
Events ... 717
TRVPrint.OnFormatting ... 717
TRVPrint.OnPagePostpaint ... 718
TRVPrint.OnPagePrepaint ... 719
TRVPrint.OnSendingToPrinter ... 719
Classes of properties ... 720
TRVVirtualPrinterProperties ... 720
Properties ... 720
TRVVirtualPrinterProperties.Active ... 721
TRVVirtualPrinterProperties.PixelsPerInch ... 721
TRVVirtualPrinterProperties.PageWidth, PageHeight ... 721
TRVReportHelper ... 721
Properties ... 722
TRVReportHelper.AllowTransparentDrawing ... 723
TRVReportHelper.TargetPixelsPerInch ... 723
TRVReportHelper.RichView ... 724
Methods ... 724
TRVReportHelper.DrawPage ... 725
TRVReportHelper.DrawPageAt ... 725
TRVReportHelper.Finished ... 726
TRVReportHelper.FormatNextPage ... 726
TRVReportHelper.GetLastPageHeight ... 726
TRVReportHelper.Init ... 727
TRVReportHelper.SavePDF ... 727
TRVReportHelper.UpdatePageNumbers ... 727
Events ... 728
TRVReportHelper.OnGetPageNumber ... 728
TRVSpellChecker ... 728
Properties ... 730
TRVSpellChecker.AvailableLanguages ... 731
TRVSpellChecker.DialogShown ... 731
TRVSpellChecker.Language ... 731
TRVSpellChecker.LanguageIndex ... 732
TRVSpellChecker.SpellFormStyle ... 732
Methods ... 732
TRVSpellChecker.AddToAutoreplaceList ... 733
TRVSpellChecker.AddToDictionary ... 733
TRVSpellChecker.AddToIgnoreList ... 733
TRVSpellChecker.CurrentLanguage ... 733
TRVSpellChecker.Execute ... 733
TRVSpellChecker.FillLanguageNames ... 734
TRVSpellChecker.GetAutoCorrection ... 734
TRVSpellChecker.GetLanguageNameFromCode ... 735
TRVSpellChecker.GuessCompletions ... 735
TRVSpellChecker.IsAvailable ... 735
TRVSpellChecker.IsWordValid ... 735
TRVSpellChecker.RegisterAsService, UnregisterAsService ... 735
TRVSpellChecker.RegisterEditor, UnregisterEditor ... 736
TRVSpellChecker.StartLiveSpelling ... 736
TRVSpellChecker.Suggest ... 737
TRVSpellChecker.SupportsFeatures ... 737
Events ... 737
TRVSpellChecker.OnSpellForm ... 737

TRVSpellChecker.OnSpellFormAction ... 738
TRVSpellChecker.OnSpellFormLocalize ... 739
TRVStyle ... 739
Properties ... 742
TRVStyle.CheckpointColor, CheckpointColor ... 743
TRVStyle.Color ... 744
TRVStyle.CurrentItemColor ... 744
TRVStyle.DefCodePage ... 745
TRVStyle.DefTabWidth ... 745
TRVStyle.DefUnicodeStyle ... 746
TRVStyle.DisabledFontColor ... 746
TRVStyle.EndnoteNumbering ... 746
TRVStyle.FieldHighlightColor ... 746
TRVStyle.FieldHighlightType ... 747
TRVStyle.FloatingLineColor ... 747
TRVStyle.FontQuality ... 748
TRVStyle.FootnotePageReset ... 749
TRVStyle.FullRedraw ... 750
TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle ... 750
TRVStyle.HoverColor ... 750
TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor, SelOpacity ... 751
TRVStyle.InvalidPicture ... 752
TRVStyle.JumpCursor ... 752
TRVStyle.LineSelectCursor ... 753
TRVStyle.LineWrapMode ... 753
TRVStyle.ListStyles ... 755
TRVStyle.LiveSpellingColor ... 755
TRVStyle.MainRVStyle ... 756
TRVStyle.PageBreakColor ... 756
TRVStyle.ParaBackAroundContent ... 757
TRVStyle.ParaStyles ... 757
TRVStyle.SelectionHandleKind ... 757
TRVStyle.SelectionMode ... 758
TRVStyle.SelectionStyle ... 759
TRVStyle.SoftPageBreakColor ... 761
TRVStyle.SpacesInTab ... 761
TRVStyle.SpecialCharactersColor ... 762
TRVStyle.StyleTemplates ... 762
TRVStyle.TextBackgroundKind ... 762
TRVStyle.TextEngine ... 763
TRVStyle.TextStyles ... 764
TRVStyle.Units ... 764
TRVStyle.UnitsPixelsPerInch ... 765
TRVStyle.UseSound ... 766
Methods ... 766
TRVStyle.AddTextStyle ... 767
TRVStyle.Create ... 767
TRVStyle.ConvertTo* ... 767
TRVStyle.DeleteTextStyle ... 767
TRVStyle.Destroy ... 767
TRVStyle.GetAs* ... 768
TRVStyle.*ToUnits ... 769
TRVStyle.GetListStyleClass ... 770
TRVStyle.GetParaStyleClass ... 770
TRVStyle.GetTextStyleClass ... 770
TRVStyle.LoadINI ... 771
TRVStyle.FindTextStyle, FindParaStyle ... 771
TRVStyle.LoadReg ... 771

TRVStyle.SaveCSS ... 772
TRVStyle.SaveCSSToStream ... 772
TRVStyle.SaveINI ... 773
TRVStyle.SaveReg ... 773
Events ... 773
TRVStyle.OnAfterApplyStyle ... 774
TRVStyle.OnApplyStyle ... 774
TRVStyle.OnApplyStyleColor ... 775
TRVStyle.OnDrawCheckpoint ... 776
TRVStyle.OnDrawPageBreak ... 778
TRVStyle.OnDrawParaBack ... 780
TRVStyle.OnDrawStyleText ... 781
TRVStyle.OnDrawTextBack ... 783
TRVStyle.OnStyleHoverSensitive ... 784
Classes of properties - Style collections ... 785
TFontInfos ... 785
Properties ... 786
TFontInfos.InvalidItem ... 786
TFontInfos.Items ... 786
TFontInfos.PixelsPerInch ... 786
Methods ... 787
TFontInfos.Add ... 787
TFontInfos.AddFont ... 787
TFontInfos.AddFontEx ... 788
TFontInfos.AssignTo ... 788
TFontInfos.Create ... 788
TFontInfos.FindStyleWithCharset ... 788
TFontInfos.FindStyleWithColor ... 788
TFontInfos.FindStyleWithFont ... 789
TFontInfos.FindStyleWithFontName ... 789
TFontInfos.FindStyleWithFontSize ... 790
TFontInfos.FindStyleWithFontStyle ... 790
TFontInfos.FindSuchStyle, FindSuchStyleEx ... 791
TParaInfos ... 791
Properties ... 792
TParaInfos.InvalidItem ... 792
TParaInfos.Items ... 792
Methods ... 793
TParaInfos.Add ... 793
TParaInfos.AssignTo ... 793
TParaInfos.Create ... 794
TParaInfos.FindStyleWithAlignment ... 794
TParaInfos.FindSuchStyle, FindSuchStyleEx ... 794
TRVListInfos ... 795
Properties ... 795
TRVListInfos.Items ... 795
Methods ... 796
TRVListInfos.FindSuchStyle ... 796
TRVListInfos.FindStyleWithLevels ... 796
TRVListInfos.AssignTo ... 797
TRVListInfos.Add ... 797
TRVStyleTemplateCollection ... 797
Properties ... 798
TRVStyleTemplateCollection.DefStyleName ... 798
TRVStyleTemplateCollection.ForbiddenIds ... 798
TRVStyleTemplateCollection.Items ... 798
TRVStyleTemplateCollection.NormalStyleTemplate ... 799
Methods ... 799

TRVStyleTemplateCollection.AssignStyleTemplates 799
TRVStyleTemplateCollection.AssignToStrings 799
TRVStyleTemplateCollection.ClearTextFormat, ClearParaFormat 800
TRVStyleTemplateCollection.FindById, FindItemById 800
TRVStyleTemplateCollection.FindByName, FindItemByName 800
TRVStyleTemplateCollection.InsertFromRVST 801
TRVStyleTemplateCollection.LoadFromRVST, SaveToRVST 801
TRVStyleTemplateCollection.LoadFromStreamRVST, SaveToStreamRVST 802
TRVStyleTemplateCollection.ResetNameCounter 802
Classes of properties - Styles 802
TCustomRVInfo, TCustomRVTextOrParaInfo 803
Properties 804
TCustomRVInfo.Standard 804
TCustomRVInfo.StyleName 805
TCustomRVFontOrParaInfo.StyleTemplateId 805
TCustomRVFontInfo 806
Properties 807
TCustomRVFontInfo.BackColor 808
TCustomRVFontInfo.BiDiMode 808
TCustomRVFontInfo.CharScale 808
TCustomRVFontInfo.Charset 809
TCustomRVFontInfo.CharSpacing 809
TCustomRVFontInfo.Color 810
TCustomRVFontInfo.EmptyWidth 810
TCustomRVFontInfo.FontName 811
TCustomRVFontInfo.HoverBackColor 811
TCustomRVFontInfo.HoverColor 812
TCustomRVFontInfo.HoverEffects 812
TCustomRVFontInfo.HoverUnderlineColor 813
TCustomRVFontInfo.JumpCursor 813
TCustomRVFontInfo.Options 814
TCustomRVFontInfo.ParaStyleTemplateId 815
TCustomRVFontInfo.Protection 815
TCustomRVFontInfo.Size, SizeDouble 817
TCustomRVFontInfo.Style 817
TCustomRVFontInfo.StyleEx 818
TCustomRVFontInfo.SubSuperScriptType 818
TCustomRVFontInfo.UnderlineColor 818
TCustomRVFontInfo.UnderlineType 819
TCustomRVFontInfo.VShift 820
Methods 820
TCustomRVFontInfo.Assign 820
TCustomRVFontInfo.AssignTo 820
TCustomRVFontInfo.Create 821
TCustomRVFontInfo.IsEqual 821
TFontInfo 821
Properties 822
TFontInfo.Jump 823
TFontInfo.ModifiedProperties 824
TFontInfo.NextStyleNo 824
TFontInfo.Unicode 824
Methods 825
TFontInfo.Assign 825
TFontInfo.Create 825
TFontInfo.Draw 825
TFontInfo.IsEqual 827
TCustomRVParaInfo 828
Properties 829

TCustomRVParaInfo.Alignment, LastLineAlignment 829
TCustomRVParaInfo.Background 830
TCustomRVParaInfo.BiDiMode 831
TCustomRVParaInfo.Border 831
TCustomRVParaInfo.FirstIndent 831
TCustomRVParaInfo.LeftIndent 832
TCustomRVParaInfo.LineSpacing 832
TCustomRVParaInfo.LineSpacingType 833
TCustomRVParaInfo.Options 833
TCustomRVParaInfo.OutlineLevel 835
TCustomRVParaInfo.RightIndent 835
TCustomRVParaInfo.SpaceAfter 835
TCustomRVParaInfo.SpaceBefore 836
TCustomRVParaInfo.Tabs 836
Methods 836
TCustomRVParaInfo.Assign 836
TCustomRVParaInfo.Create 836
TCustomRVParaInfo.IsEqual 837
TParaInfo 837
Properties 838
TParaInfo.DefStyleNo 839
TParaInfo.ModifiedProperties 839
TParaInfo.NextParaNo 840
Methods 840
TParaInfo.Assign 840
TParaInfo.Create 840
TParaInfo.IsEqual 841
TRVListInfo 841
Properties 842
TRVListInfo.Levels 842
TRVListInfo.OneLevelPreview 842
Methods 842
TRVListInfo.AllNumbered 842
TRVListInfo.HasNumbering 843
TRVStyleTemplate 843
Properties 844
TRVStyleTemplate.Id 844
TRVStyleTemplate.Kind 845
TRVStyleTemplate.Name 845
TRVStyleTemplate.NextId, Next 846
TRVStyleTemplate.ParaStyle 846
TRVStyleTemplate.ParentId, Parent 847
TRVStyleTemplate.QuickAccess 847
TRVStyleTemplate.TextStyle 847
TRVStyleTemplate.ValidParaProperties 847
TRVStyleTemplate.ValidTextProperties 848
Methods 848
TRVStyleTemplate.ApplyToParaStyle 848
TRVStyleTemplate.ApplyToTextStyle 849
TRVStyleTemplate.UpdateModifiedTextStyleProperties, UpdateModifiedParaStyleProperties 850
Classes of properties of a paragraph style 851
TRVBackgroundRect 851
Properties 851
TRVBackgroundRect.BorderOffsets 851
TRVBackgroundRect.Color 852
TRVBackgroundRect.Opacity 852
TRVBorder 852
Properties 853

TRVBorder.BorderOffsets ... 853
TRVBorder.Color ... 853
TRVBorder.InternalWidth ... 853
TRVBorder.Style ... 854
TRVBorder.VisibleBorders ... 854
TRVBorder.Width ... 854
TRVTabInfos ... 854
Properties ... 855
TRVTabInfos.Items ... 855
Methods ... 855
TRVTabInfos.AddFrom ... 855
TRVTabInfos.DeleteList ... 855
TRVTabInfos.Find ... 855
TRVTabInfos.Intersect ... 856
TRVTabInfo ... 856
Properties ... 856
TRVTabInfo.Align ... 856
TRVTabInfo.Leader ... 857
TRVTabInfo.Position ... 857
Classes of properties of a list style ... 858
TRVListLevelCollection ... 858
Properties ... 858
TRVListLevelCollection.Items ... 859
Methods ... 859
TRVListLevelCollection.Add ... 859
TRVListLevel ... 859
Properties ... 860
TRVListLevel.FirstIndent ... 860
TRVListLevel.Font ... 861
TRVListLevel.FormatString ... 861
TRVListLevel.FormatStringW ... 862
TRVListLevel.ImageIndex ... 863
TRVListLevel.ImageList ... 863
TRVListLevel.ImageWidth, ImageHeight ... 863
TRVListLevel.LeftIndent ... 864
TRVListLevel.ListType ... 864
TRVListLevel.MarkerAlignment ... 866
TRVListLevel.MarkerIndent ... 866
TRVListLevel.Options ... 867
TRVListLevel.Picture ... 868
TRVListLevel.StartFrom ... 868
Methods ... 868
TRVListLevel.HasNumbering ... 868
3 Components ancestor classes ... 868
TRVScroller ... 869
Properties ... 870
TRVScroller.BorderStyle ... 871
TRVScroller.HScrollMax ... 871
TRVScroller.HScrollPos ... 872
TRVScroller.HScrollVisible ... 872
TRVScroller.InplaceEditor ... 872
TRVScroller.NoVScroll ... 873
TRVScroller.Tracking ... 873
TRVScroller.UseXPThemes ... 873
TRVScroller.VScrollMax ... 874
TRVScroller.VScrollPos ... 874
TRVScroller.VScrollVisible ... 874

TRVScroller.WheelStep ... 875
Methods ... 875
TRVScroller.Create ... 875
TRVScroller.ScrollTo[Position] ... 875
Events ... 876
TRVScroller.OnHScrolled ... 877
TRVScroller.OnVScrolled ... 877
TCustomRVPrint ... 877
Properties ... 877
TCustomRVPrint.ColorMode ... 878
TCustomRVPrint.DarkMode ... 878
TCustomRVPrint.EndAt ... 878
TCustomRVPrint.FirstPageNo ... 879
TCustomRVPrint.IgnorePageBreaks ... 879
TCustomRVPrint.LastPageNo ... 879
TCustomRVPrint.MaxPrintedItemNo ... 879
TCustomRVPrint.MetafileCompatibility ... 880
TCustomRVPrint.MetafilePixelsPerInch ... 880
TCustomRVPrint.MinPrintedItemNo ... 881
TCustomRVPrint.NoMetafiles ... 881
TCustomRVPrint.PageNoFromNumber ... 881
TCustomRVPrint.PagesCount ... 882
TCustomRVPrint.PreviewCorrection ... 882
TCustomRVPrint.StartAt ... 882
TCustomRVPrint.TransparentBackground ... 882
Methods ... 882
TCustomRVPrint.CanSavePDF ... 883
TCustomRVPrint.Clear ... 883
TCustomRVPrint.GetFirstItemOnPage ... 883
TCustomRVPrint.GetPageNo ... 883
TCustomRVPrint.UpdatePaletteInfo ... 884
Events ... 885
TCustomRVPrint.OnAfterPrintImage ... 885
TCustomRVPrint.OnDrawCheckpoint ... 885
TCustomRVPrint.OnDrawHyperlink ... 886
TCustomRVPrint.OnPrintComponent ... 887
TCustomRVPrintPreview ... 888
Properties ... 889
TCustomRVPrintPreview.ClickMode ... 891
TCustomRVPrintPreview.Color, Fill ... 891
TCustomRVPrintPreview.DarkModeUI ... 892
TCustomRVPrintPreview.MarginsPen, MarginsStroke ... 892
TCustomRVPrintPreview.PageBorderColor ... 893
TCustomRVPrintPreview.PageBorderWidth ... 893
TCustomRVPrintPreview.PageNo ... 893
TCustomRVPrintPreview.PrintableAreaPen, PrintableAreaStroke ... 893
TCustomRVPrintPreview.ShadowColor ... 894
TCustomRVPrintPreview.ShadowWidth ... 894
TCustomRVPrintPreview.ZoomInCursor ... 895
TCustomRVPrintPreview.ZoomMode ... 895
TCustomRVPrintPreview.ZoomOutCursor ... 896
TCustomRVPrintPreview.ZoomPercent ... 896
Methods ... 896
TCustomRVPrintPreview.Create ... 897
TCustomRVPrintPreview.First ... 897
TCustomRVPrintPreview.Last ... 897
TCustomRVPrintPreview.Next ... 897
TCustomRVPrintPreview.Prev ... 898

TCustomRVPrintPreview.SetZoom ... 898
Events ... 898
TCustomRVPrintPreview.OnZoomChanged ... 899

Part VII Item types 899

1 Table - TRVTableItemInfo ... 901

Properties ... 903
TRVTableItemInfo.BackgroundImage ... 904
TRVTableItemInfo.BackgroundImageFileName ... 905
TRVTableItemInfo.BackgroundStyle ... 905
TRVTableItemInfo.BackgroundProperties ... 905
TRVTableItemInfo.BestWidth ... 906
TRVTableItemInfo.BorderColor ... 906
TRVTableItemInfo.BorderHSpacing ... 907
TRVTableItemInfo.BorderLightColor ... 907
TRVTableItemInfo.BorderStyle ... 908
TRVTableItemInfo.BorderVSpacing ... 908
TRVTableItemInfo.BorderWidth ... 909
TRVTableItemInfo.CellBorderColor ... 909
TRVTableItemInfo.CellBorderLightColor ... 910
TRVTableItemInfo.CellBorderStyle ... 910
TRVTableItemInfo.CellBorderWidth ... 910
TRVTableItemInfo.CellHPadding ... 911
TRVTableItemInfo.CellHSpacing ... 911
TRVTableItemInfo.CellOverrideColor ... 912
TRVTableItemInfo.CellPadding ... 912
TRVTableItemInfo.Cells ... 912
TRVTableItemInfo.CellVPadding ... 913
TRVTableItemInfo.CellVSpacing ... 913
TRVTableItemInfo.ColBandSize, RowBandSize ... 913
TRVTableItemInfo.ColCount ... 914
TRVTableItemInfo.Color ... 914
TRVTableItemInfo.HeadingRowCount ... 914
TRVTableItemInfo.HOutermostRule ... 915
TRVTableItemInfo.HRuleColor ... 915
TRVTableItemInfo.HRuleWidth ... 916
TRVTableItemInfo.Opacity ... 916
TRVTableItemInfo.Options ... 916
TRVTableItemInfo.PrintOptions ... 919
TRVTableItemInfo.RowCount ... 920
TRVTableItemInfo.Rows ... 920
TRVTableItemInfo.TextColSeparator ... 920
TRVTableItemInfo.TextRowSeparator ... 921
TRVTableItemInfo.VisibleBorders ... 921
TRVTableItemInfo.VOutermostRule ... 921
TRVTableItemInfo.VRuleColor ... 922
TRVTableItemInfo.VRuleWidth ... 922
TRVTableItemInfo's row and column colors ... 923
Methods ... 924
TRVTableItemInfo.AssignProperties ... 925
TRVTableItemInfo.CanMergeCells ... 925
TRVTableItemInfo.CanMergeSelectedCells ... 926
TRVTableItemInfo.Changed ... 927
TRVTableItemInfo.Create ... 927
TRVTableItemInfo.CreateEx ... 928
TRVTableItemInfo.DeleteCols ... 928
TRVTableItemInfo.DeleteEmptyCols ... 929

TRVTableItemInfo.DeleteEmptyRows ... 929
TRVTableItemInfo.DeleteRows ... 929
TRVTableItemInfo.DeleteSelectedCols ... 930
TRVTableItemInfo.DeleteSelectedRows ... 930
TRVTableItemInfo.Deselect ... 931
TRVTableItemInfo.Destroy ... 931
TRVTableItemInfo.EditCell ... 931
TRVTableItemInfo.GetCellAt ... 932
TRVTableItemInfo.GetEditedCell ... 932
TRVTableItemInfo.GetNormalizedSelectionBounds ... 933
TRVTableItemInfo.GetSelectionBounds ... 933
TRVTableItemInfo.InsertCols ... 934
TRVTableItemInfo.InsertColsLeft ... 934
TRVTableItemInfo.InsertColsRight ... 935
TRVTableItemInfo.InsertRows ... 935
TRVTableItemInfo.InsertRowsAbove ... 936
TRVTableItemInfo.InsertRowsBelow ... 936
TRVTableItemInfo.IsCellSelected ... 936
TRVTableItemInfo.LoadFromStream ... 937
TRVTableItemInfo.MergeCells ... 937
TRVTableItemInfo.MergeSelectedCells ... 938
TRVTableItemInfo.MoveRows ... 938
TRVTableItemInfo.SaveRowsToStream ... 939
TRVTableItemInfo.SaveToStream ... 939
TRVTableItemInfo.Select ... 939
TRVTableItemInfo.SelectCols ... 940
TRVTableItemInfo.SelectRows ... 940
TRVTableItemInfo.SetBackgroundProperty ... 940
TRVTableItemInfo.SetCellBackgroundImage ... 941
TRVTableItemInfo.SetCellBackgroundImageFileName ... 941
TRVTableItemInfo.SetCellBackgroundProperty ... 941
TRVTableItemInfo.SetCellBackgroundStyle ... 942
TRVTableItemInfo.SetCellBestHeight ... 942
TRVTableItemInfo.SetCellBestWidth ... 942
TRVTableItemInfo.SetCellBorderColor ... 942
TRVTableItemInfo.SetCellBorderLightColor ... 943
TRVTableItemInfo.SetCellColor ... 943
TRVTableItemInfo.SetCellHint ... 943
TRVTableItemInfo.SetCellOpacity ... 943
TRVTableItemInfo.SetCellOptions ... 944
TRVTableItemInfo.SetCellRotation ... 944
TRVTableItemInfo.SetCellTag ... 944
TRVTableItemInfo.SetCellVAlign ... 945
TRVTableItemInfo.SetCellVisibleBorders ... 945
TRVTableItemInfo.SetRow* ... 945
TRVTableItemInfo.SetTableVisibleBorders ... 946
TRVTableItemInfo.SplitSelectedCellsHorizontally ... 946
TRVTableItemInfo.SplitSelectedCellsVertically ... 946
TRVTableItemInfo.UnmergeCells ... 947
TRVTableItemInfo.UnmergeSelectedCells ... 947
Events ... 948
TRVTableItemInfo.OnCellEditing ... 948
TRVTableItemInfo.OnDrawBackground ... 949
TRVTableItemInfo.OnDrawBorder ... 949
Classes of properties ... 951
TRVTableCellData ... 951
Properties ... 952
TRVTableCellData.BackgroundImage ... 953

TRVTableCellData.BackgroundImageFileName 953
TRVTableCellData.BackgroundStyle 954
TRVTableCellData.BackgroundProperties 954
TRVTableCellData.BestHeight 955
TRVTableCellData.BestWidth 955
TRVTableCellData.BorderColor 956
TRVTableCellData.BorderLightColor 956
TRVTableCellData.Color 957
TRVTableCellData.ColSpan 957
TRVTableCellData.Hint 958
TRVTableCellData.IgnoreContentHeight 958
TRVTableCellData.Opacity 958
TRVTableCellData.Options 959
TRVTableCellData.Rotation 961
TRVTableCellData.RowSpan 962
TRVTableCellData.Tag 963
TRVTableCellData.VAlign 963
TRVTableCellData.VisibleBorders 963
Methods 964
TRVTableCellData.BackgroundPropertiesAllocate, BackgroundPropertiesFinalize 964
TRVTableCellData.GetRVData 964
Additional properties of TRVTableCellData 965
TRVTableRow 966
Properties 967
TRVTableRow.Items 967
TRVTableRow.KeepTogether 967
TRVTableRow.PageBreakBefore 967
TRVTableRow.VAlign 968
Methods 968
TRVTableRows 968
Properties 969
TRVTableRows.Items 969
Methods 969
TRVTableRows.GetMainCell 969
2 Label - TRVLabelItemInfo 970
Properties 971
TRVLabelItemInfo.Alignment 971
TRVLabelItemInfo.Cursor 971
TRVLabelItemInfo.MinWidth 971
TRVLabelItemInfo.ProtectTextStyleNo 972
TRVLabelItemInfo.RemoveInternalLeading, RemoveSpaceBelow 972
TRVLabelItemInfo.Text 972
TRVLabelItemInfo.TextStyleNo 973
Methods 973
TRVLabelItemInfo.Create 973
TRVLabelItemInfo.CreateEx 973
3 Math expression - TRVMathItemInfo 974
Properties 975
TRVMathItemInfo.DisplayInline 975
TRVMathItemInfo.FontName 976
TRVMathItemInfo.FontSizeDouble 976
TRVMathItemInfo.Text 977
TRVMathItemInfo.TextColor 977
Methods 977
TRVMathItemInfo.Create 978
4 Numbered sequence - TRVSeqItemInfo 978
Properties 979

TRVSeqItemInfo.FormatString 980
TRVSeqItemInfo.NumberType 980
TRVSeqItemInfo.Reset 981
TRVSeqItemInfo.SeqName 981
TRVSeqItemInfo.StartFrom 981
Methods 981
TRVSeqItemInfo.Create 982
TRVSeqItemInfo.CreateEx 982
5 Ancestor for notes - TCustomRVNoteItemInfo 982
Properties 983
TCustomRVNoteItemInfo.Document 984
TCustomRVNoteItemInfo.NoteText 984
Methods 984
TCustomRVNoteItemInfo.CreateEx 984
TCustomRVNoteItemInfo.ReplaceDocumentEd 985
6 Endnote - TRVEndnoteItemInfo 985
Properties 986
Methods 986
TRVEndnoteItemInfo.Create 987
TRVEndnoteItemInfo.CreateEx 987
7 Footnote - TRVFootnoteItemInfo 988
Properties 989
Methods 989
TRVFootnoteItemInfo.Create 990
TRVFootnoteItemInfo.CreateEx 990
8 Sidenote - TRVSidenoteItemInfo 990
Properties 992
TRVSidenoteItemInfo.BoxProperties 992
TRVSidenoteItemInfo.BoxPosition 993
Methods 993
TRVSidenoteItemInfo.Create 993
TRVSidenoteItemInfo.CreateEx 993
Classes of properties 994
TRVBoxProperties 994
Properties 994
TRVBoxProperties.Background 995
TRVBoxProperties.Border 995
TRVBoxProperties.Height, HeightType 995
TRVBoxProperties.VAlign 996
TRVBoxProperties.Width, WidthType 996
TRVBoxPosition 997
Properties 998
TRVBoxPosition.HorizontalAnchor 998
TRVBoxPosition.HorizontalXXX 999
TRVBoxPosition.PositionInText 1001
TRVBoxPosition.RelativeToCell 1002
TRVBoxPosition.VerticalAnchor 1002
TRVBoxPosition.VerticalXXX 1003
9 Text box - TRVTextBoxItemInfo 1004
Methods 1005
TRVTextBoxItemInfo.Create 1005
TRVTextBoxItemInfo.CreateEx 1005
10 Reference to the parent note - TRVNoteReferenceItemInfo 1006
Properties 1007
Methods 1008
TRVNoteReferenceItemInfo.Create 1008

TRVNoteReferenceItemInfo.CreateEx ... 1008
11 Ancestor for text fields - TCustomRVFieldItemInfo ... 1009
Properties ... 1009
TCustomRVPageNumberItemInfo.NumberType ... 1009
12 Page Number - TRVPageNumberItemInfo ... 1010
Methods ... 1010
TRVPageNumberItemInfo.Create ... 1010
TRVPageNumberItemInfo.CreateEx ... 1011
13 Page Count - TRVPageCountItemInfo ... 1011
Methods ... 1012
TRVPageCountItemInfo.Create ... 1012
TRVPageCountItemInfo.CreateEx ... 1012
Part VIII Other classes 1013
1 ERichViewError ... 1014
2 TCustomRVData and others (TRichView documents) ... 1014
TCustomRVData.Edit ... 1016
TCustomRVData.GetRVData ... 1017
TCustomRVData.GetSourceRVData ... 1017
TCustomRVFormattedData.GetOriginEx ... 1018
3 TPrintableRV ... 1018
4 TRVBooleanRect ... 1018
5 TRVCanvas ... 1018
6 TRVControlsPainter ... 1019
7 TRVCustomMathEquationManager ... 1020
Properties ... 1020
TRVCustomMathEquationManager.SaveMathMLInHTML ... 1021
TRVCustomMathEquationManager.SaveOMMLInDocX ... 1021
8 TRVBackgroundProperties ... 1021
Properties ... 1022
TRVBackgroundProperties.Attachment ... 1022
TRVBackgroundProperties.HorizontalAlignment ... 1023
TRVBackgroundProperties.HorizontalOffset, HorizontalOffsetKind ... 1023
TRVBackgroundProperties.HorizontalRepeat ... 1024
TRVBackgroundProperties.HorizontalSizeKind, ImageWidth ... 1024
TRVBackgroundProperties.VerticalAlignment ... 1024
TRVBackgroundProperties.VerticalOffset, VerticalOffsetKind ... 1025
TRVBackgroundProperties.VerticalRepeat ... 1025
TRVBackgroundProperties.VerticalSizeKind, ImageHeight ... 1025
TRVBackgroundProperties.Visible ... 1025
9 TRVDefaultLoadProperties ... 1026
Properties ... 1026
TRVDefaultLoadProperties.DefaultBulletFontName ... 1026
TRVDefaultLoadProperties.DefaultFontName ... 1026
TRVDefaultLoadProperties.DefaultFontSizeDouble ... 1027
TRVDefaultLoadProperties.DefaultFontSizesDouble ... 1027
TRVDefaultLoadProperties.DefaultMonoSpacedFontName ... 1028
TRVDefaultLoadProperties.DefaultProperties ... 1028
TRVDefaultLoadProperties.DefaultUnicodeSymbolFontName ... 1028
TRVDefaultLoadProperties.GenericFontNames ... 1028
TRVDefaultLoadProperties.ListMarkerIndentPix ... 1031
10 TRVDeleteUnusedStylesData ... 1031

11 TRVFont ... 1032
12 TRVGraphic ... 1032
13 TRVGraphicClass ... 1033
14 TRVGraphicHandler ... 1033
15 TRVIntegerList ... 1036
16 TRVItemFactory ... 1036
17 TRVMathDocObject ... 1037
18 TRVPicture ... 1037
19 TRVRect ... 1038
20 TRVReportHelperWithHeaderFooters ... 1038
Properties ... 1039
TRVReportHelperWithHeaderFooters.MainDoc ... 1039
TRVReportHelperWithHeaderFooters.UseStyleTemplates ... 1039
Methods ... 1039
TRVReportHelperWithHeaderFooters.AssignToRVPrint ... 1040
TRVReportHelperWithHeaderFooters.Create ... 1040
TRVReportHelperWithHeaderFooters.GetFooter ... 1040
TRVReportHelperWithHeaderFooters.GetHeader ... 1040
TRVReportHelperWithHeaderFooters.Reset ... 1041
21 TRVStringList ... 1041
22 TRVThumbnailMaker ... 1041
23 TRVUnitsRect ... 1042

Part IX Procedures and functions 1043

1 DrawControl and Others ... 1043
2 Functions for TRVUnits conversion ... 1044
3 Functions from RVGetText Unit ... 1045
4 Functions from RVLinear Unit ... 1047
5 Functions from RVMathItem Unit ... 1048
6 GetRVESearchOptions ... 1049
7 GetRVSearchOptions ... 1049
8 HTML from RTF de-encapsulation ... 1049
9 RV_GetPrinterDC ... 1050
10 RVGetFirstEndnote and Others ... 1052
11 RVGetNoteTextStyleNo ... 1052
12 RVIsURL, RVIsEmail ... 1053
13 RVMathEquationManager ... 1054
14 RVItemFactory ... 1054
15 RVOpenURL ... 1054
16 Unicode Conversion ... 1054

Part X Types 1055

1 TCheckpointData ... 1055
2 TRVAnsiString ... 1056
3 TRVBackgroundPropertyId ... 1056

4. TRVBackgroundRepeat 1057
5. TRVBackgroundSize 1058
6. TRVBiDiMode 1058
7. TRVBorderStyle 1059
8. TRVBreakStyle 1059
9. TRVCellVAlign 1061
10. TRVChatFileType 1061
11. TRVChatItemId 1061
12. TRVChatMessageState 1062
13. TRVChatQuoteType 1062
14. TRVCodePage 1062
15. TRVColor 1063
16. TRVColorMode 1063
17. TRVCoord, TRVCoordPoint, TRVCoordSize, TRVCoordRect 1064
18. TRVDBFieldFormat 1064
19. TRVDisplayOptions 1065
20. TRVDocRotation 1065
21. TRVDualLength, TRVDualSignedLength, TRVDualSignedLengthKind 1066
22. TRVEnumScope 1066
23. TRVESearchOptions 1066
24. TRVExtraItemProperty 1067
25. TRVExtraItemStrProperty 1072
26. TRVFloatPosition 1073
27. TRVFloatPositionKind 1073
28. TRVFloatProperty 1074
29. TRVFloatSize 1074
30. TRVFontCharset 1075
31. TRVFontInfoProperties 1075
32. TRVFontSize 1076
33. TRVFReaderStyleMode 1076
34. TRVGraphicType 1077
35. TRVHFType 1078
36. TRVHTMLEncoding 1078
37. TRVHTMLLength 1079
38. TRVHTMLSavingPart 1079
39. TRVIntProperty 1080
40. TRVItemBackgroundStyle 1080
41. TRVLength 1081
42. TRVLoadFormat 1081
43. TRVOleDropEffects 1081
44. TRVOpacity 1082
45. TRVPageNumberType 1082

46 TRVPageSet ... 1083
47 TRVParaInfoProperties ... 1083
48 TRVPDFMetadata ... 1084
49 TRVPenStyle ... 1084
50 TRVPixel96Length ... 1085
51 TRVPixelLength ... 1085
52 TRVPrintingStep ... 1085
53 TRVRawByteString ... 1085
54 TRVReaderStyleMode ... 1086
55 TRVSaveFormat ... 1086
56 TRVSaveOptions ... 1086
57 TRVScreenAndDevice, PRVScreenAndDevice ... 1090
58 TRVSearchOptions ... 1091
59 TRVSeqType ... 1092
60 TRVSpellFormAction ... 1093
61 TRVStrProperty ... 1093
62 TRVStyleLength ... 1094
63 TRVStyleTemplateId ... 1094
64 TRVStyleTemplateName ... 1094
65 TRVStyleUnits ... 1094
66 TRVTableBorderStyle ... 1096
67 TRVTag ... 1096
68 TRVTextDrawStates ... 1097
69 TRVUndoType ... 1098
70 TRVUnicodeString ... 1099
71 TRVUnits ... 1099
72 TRVVAlign ... 1100
73 TRVYesNoAuto ... 1101
74 TRVZoomPercent ... 1102

Part XI Global constants and variables 1102

1 Clipboard Related Constants and Variables ... 1104
2 Color constants ... 1105
3 Constants Related to HTML and Text ... 1107
4 Constants Related to the Caret ... 1108
5 DefaultRVFPixelsPerInch ... 1109
6 Pen style constants ... 1110
7 RichViewAllowCopyTableCells ... 1110
8 RichViewAlternativePicPrint ... 1110
9 RichViewApostropheInWord ... 1111
10 RichViewAutoAssignNormalStyleTemplate ... 1111
11 RichViewCompareStyleNames ... 1111

12 RichViewCSSUseSystemColors ... 1112
13 RichViewCtrlUpDownScroll ... 1112
14 RichViewDoNotCheckRVFStyleRefs ... 1112
15 RichViewDoNotMergeNumbering ... 1113
16 RichViewEditDefaultProportionalResize ... 1113
17 RichViewEditEnterAllowsEmptyMarkeredLines ... 1113
18 RichViewJustifyBeforeLineBreak ... 1114
19 RichViewMaxPictureCount ... 1114
20 RichViewMaxThumbnailCount ... 1114
21 RichViewPixelsPerInch ... 1115
22 RichViewRemoveNbSpAfterImport ... 1115
23 RichViewResetStandardFlag ... 1115
24 RichViewSaveHyperlinksInDocXAsFields ... 1116
25 RichViewShowGhostSpaces ... 1116
26 RichViewStringFormatMethod ... 1116
27 RichViewTableAutoAddRow ... 1117
28 RichViewTableDefaultRTFAutofit ... 1117
29 RichViewUnicodeInput ... 1118
30 RV_CP_DEFAULT ... 1118
31 RVAlwaysShowPlaceholders ... 1119
32 RVControlsPainter ... 1119
33 RVDefaultLoadProperties ... 1119
34 rveipc*** constants ... 1119
35 RVEMPTYTAG ... 1124
36 rvespc*** constants ... 1124
37 RVGetSystemColor ... 1125
38 RVGraphicHandler ... 1125
39 RVIsCustomURL ... 1126
40 RVParagraphMarks ... 1126
41 rvs*** constants ... 1127
42 RVThumbnailMaker ... 1129
43 RVVisibleSpecialCharacters ... 1129

Part XII How to... 1130

1 change page size and orientation ... 1131
2 draw page numbers, headers and footers ... 1131
3 implement commands like "make bold", "apply font", etc. ... 1131
4 implement Find and Replace dialogs ... 1132
5 switch insert/overtype mode ... 1132
6 move the caret to the beginning or to the end of document in editor ... 1133
7 make Unicode editor ... 1134
8 make a plain text editor ... 1134

9 implement smart indenting 1136
10 combine several TRichView documents in one 1136
11 use third-party graphic classes with TRichView 1136
12 create a chat window 1139
13 remove text formatting 1139

Index 1142

1 TRichView reference

Copyright © TRichView.com (www.trichview.com)

Version 24 (what's new? 38)

VCL/FireMonkey 161/LCL 160 Components

Main Components

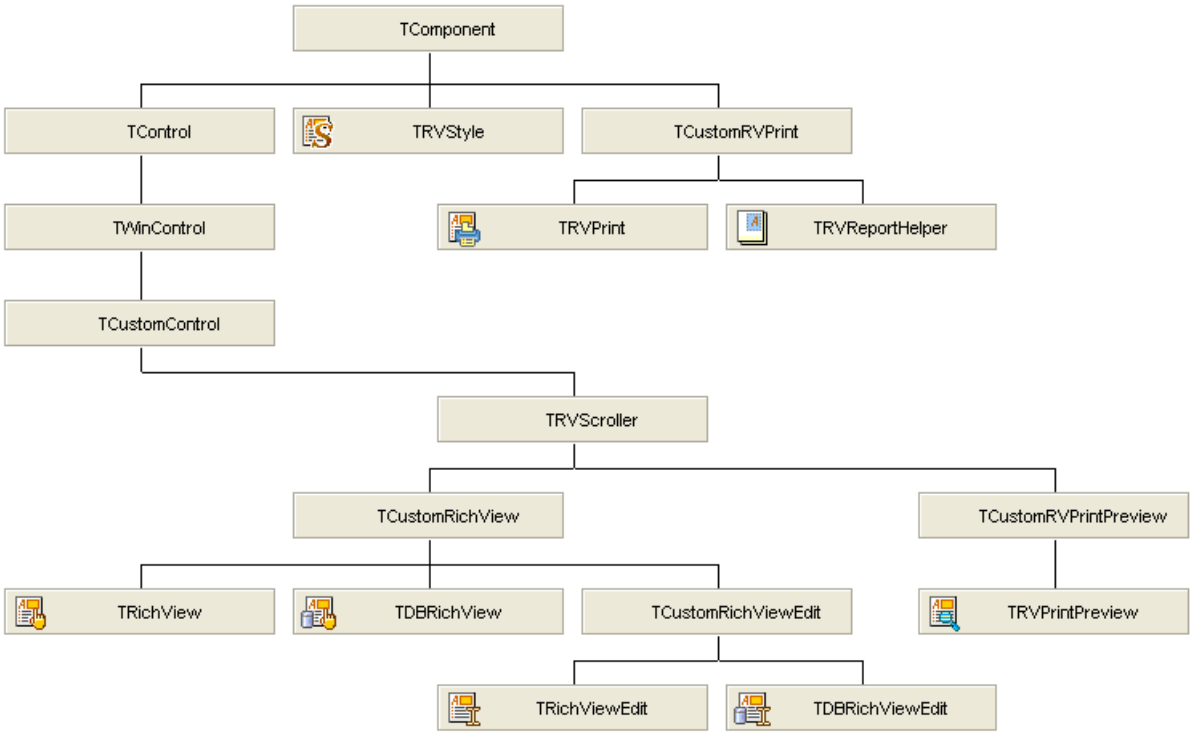
Icon	Class Name	Description
	TRVStyle 739	Component for defining visual appearance of TRichView 216, TRichViewEdit 616, TDBRichView 604, TDBRichViewEdit 616 controls. It contains some properties which can be used by several TRichView controls.
	TRichView 216	Component for displaying rich text documents
	TRichViewEdit 469	Component for editing rich text documents
	TRVPrint 696	Component for printing documents contained in TRichView 216, TRichViewEdit 616, TDBRichView 604, TDBRichViewEdit 616
	TRVPrintPreview 635	Component allowing to show document as it will be printed by TRVPrint
	TRVReportHelper 721	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter 687	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL; obsolete]
	TRVSpellChecker 728	Component for implementing spelling checking in TRichView 216, TRichViewEdit 616, TDBRichView 604, TDBRichViewEdit 616, TSRichViewEdit, TDBSRichViewEdit controls
	TRVChat 640	A component for implementing a chat in TRichView 216.

Data-Aware Components [VCL and LCL]

Icon	Class Name	Description
	TDBRichView 604	Data-aware version of TRichView [VCL and LCL]
	TDBRichViewEdit 616	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVDataSourceLink 685	Component allowing assigning TDataSource component to controls inserted in TRichView 216, TRichViewEdit 616, TDBRichView 604, TDBRichViewEdit 616, TSRichViewEdit, TDBSRichViewEdit controls

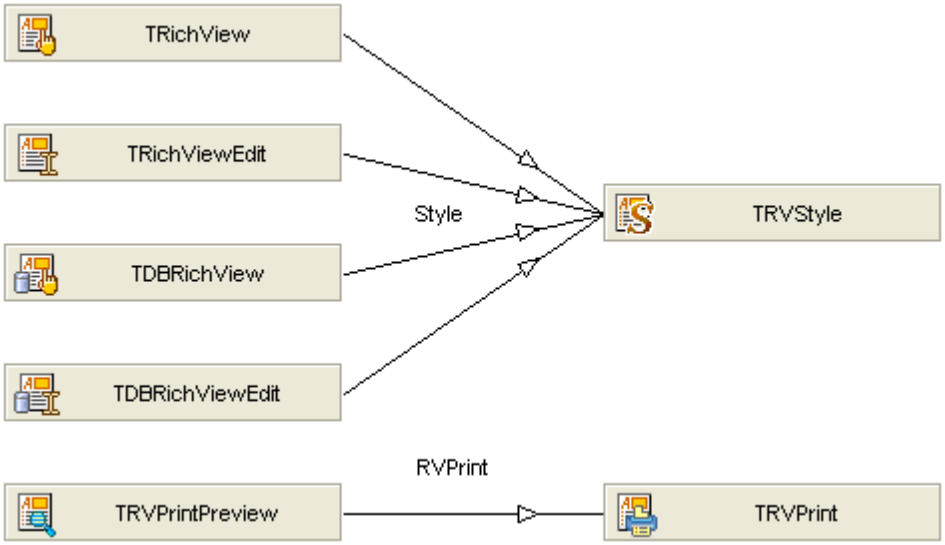
		[VCL and LCL]

Class Hierarchy (VCL and LCL)



Hierarchy of TRichView classes

Diagram of Using



How TRichView classes use each other

Related topics

- version history and compatibility [38]

- overview of features [89]
- overview of item types in documents [163]
- overview of tables [195]
- how to... [1130]

2 Version history

Version History and Compatibility Issues

- New in version 24 [38] (TRVChat, BackgroundProperties, Markdown in Clipboard)
- New in version 23 [40] (TRVSpellChecker, colors, Lazarus for Linux, equations in files)
- New in version 22 [42] (FMX for Linux and iOS, decimal tabs, Skia4Delphi)
- New in version 21 [43] (HTML import, reworked HTML export, LiveBindings, FMX for Android, update checker)
- New in version 20 [45] (Markdown import, FMX for Windows and macOS)
- New in version 19 [47] (DocX import, Markdown export, TRVDataSourceLink)
- New in version 18 [48] (Lazarus for Windows, all-Unicode, High DPI, EMU)
- New in version 17 [51] (TRVMathItemInfo, row/column bands, header/footer)
- New in version 16 [54] (Uniscribe, improved bi-di, vertical text, "distribute" alignment, semi-transparency)
- New in version 15 [56] (DocX export, text boxes and sidenotes, touch screen)
- New in version 14 [58] (StyleTemplates, cell rotation, 64-bit, VCL styles)
- New in version 13 [62] (new line wrapping, hidden text, twips)
- New in version 12 [64] (left- and right-aligned objects, headings, showing more special characters)
- New in version 11 [65] (Unicode)
- New in version 10 [68] (animations, smart pop-ups)
- New in version 1.9 [72] (tabs, live spelling check, showing special characters)
- New in version 1.8 [75] (drag and drop, table background, mouse resizing)
- New in version 1.7 [77] (bullets and numbering)
- New in version 1.6 [80] (TRVOfficeConverter, TRVReportHelper)
- New in version 1.5 [82] (RTF import, storing styles in RVF, hypertext pictures)
- New in version 1.4 [85] (tables)
- New in version 1.3 [86] (undo/redo, Unicode)
- New in version 1.2 [87] (paragraph borders and background, page breaks, search upwards)

2.1 New in version 24

Compatibility issues

The class variable RichViewTextItemClass has been removed and replaced with RVItemFactory [1054] .TextItemClass.

The background image and its properties are stored in RVF differently. Older versions of TRichView will not be able to read these parameters from RVF (they will be skipped). The new version can both read and write both representations of the background in RVF.

RAD Studio 13 Florence

Delphi and C++Builder 13 are supported.

Chat

New component: TRVChat[640], a component for implementing a chat window in TRichView[216].

Background

The background properties of TRichView are now defined in a new way. Instead of TRichView.BackgroundBitmap[229], you should use BackgroundPicture[229], which can contain any image type supported in Delphi. Instead of TRichView.BackgroundStyle[229], you should now use BackgroundProperties[229], which specify the size, position, and repetition of the background image in a much more flexible way.

Similar changes have been made for tables as well: Table.BackgroundProperties[905] instead of BackgroundStyle[905]; Cell.BackgroundProperties[954] instead of BackgroundStyle[954].

The methods have been added for assigning values to properties as editing operations (with undo support): TRichViewEdit.SetBackgroundPropertyEd[557], Table.SetCellBackgroundProperty[941], Table.SetBackgroundProperty[940].

The old properties are supported for backward compatibility.

Saving and loading, Clipboard

New methods: TRichView.CopyMarkdown[289] and TRichViewEdit.PasteMarkdown[546]. These methods are alternatives to CopyText and PasteText: they also copy/paste a plain text, but as Markdown.

New option for suppressing underline for hyperlinks around images in HTML: *rvhtmlsioSuppressLinkUnderline* in TRichView.HTMLSaveProperties[244].ImageOptions[442]. It is useful [only] for compatibility with Microsoft Outlook.

Other changes

New option *rvteHighlightable* in TRVStyle.TextStyles[].Options[814] allows applying HoverColor[812], HoverBackColor[811], HoverEffects[812] properties for non-hypertext styles.

New property TRVStyle.ParaBackAroundContent[757] specifies how boundaries of a paragraphs' background areas are determined.

TRVSpellChecker[728] supports IFMXSpellCheckerExService when working as a FireMonkey platform service in Delphi 13+.

2.2 New in version 23

This topic included changes introduced in v23.1

Compatibility issues

- New parameter (DarkMode) in TRVStyle.OnDrawStyleText[781]. New parameters (Printing, ColorMode, DarkMode) in TRVStyle.OnApplyStyleColor[775]. If you use these events, you need to correct declarations of event handlers accordingly.
- New parameter (DarkMode) in TFontInfo.Draw[825].
- If the commercial version of Adit Math Engine is used, equation objects[193] are saved in DocX and HTML as math objects, not as images.
- When reading a circle bullet from HTML or Markdown, TRichView uses Unicode white circle character (instead of "o" character of "Courier New" font)
- The package structure for Lazarus has been changed, see below.

Spelling Check

New component: TRVSpellChecker[728]. It performs spelling check in Windows, macOS, iOS, Linux (Linux version needs Hunspell). In FireMonkey, it can be used as a platform service (to check spelling in TEdit and TMemo).

Lazarus

Linux

TRichView supports Linux platform in Lazarus[160] (GTK2 widget set + Cairo canvas for printing).

New package structure

The package structure for Lazarus has been changed so that each package is in its own folder. It is especially important for Linux version of Lazarus.

First, the separation of runtime and designtime packages for Lazarus has been removed: instead of the two packages, *rvpkglaz.dpk* (runtime) and *rvpkglaz_dsgn.lpk* (designtime), there is now a single package, *rvpkglaz.dpk* (runtime + designtime). These changes have been applied to all Lazarus packages.

Second, the package with DB components (*rvdbpkg.laz*) has been removed, and its units have been included in the main package.

Third, the GDI+ support package for Lazarus has been moved to a separate folder: *Lazarus\GDIPlus\Source*.

Instead of the package group *TRichViewLazarus.lpg*, there are now two package groups: *TRichViewLazarusWin.lpg* (main packages) and *TRichViewLazarusLin.lpg* (main packages supported in Linux).

If you installed **laz_dsgn.lpk* packages of previous versions of our components, uninstall them.

If you used components from *rvdbpkg_laz.dpk* in your project, remove *rvdbpkglaz* from the project dependencies, and add *rvpkglaz* instead.

Appearance

New properties of TRichView:

- DarkMode[234] inverts luminance of colors
- UseFMXThemes[263] [FMX] allows using text color from FireMonkey style.

New property of TRVPrint and TRVReportHelper:

- DarkMode[878] inverts luminance of colors (when drawing on a canvas or printing)

New property of TRVPrintPreview:

- DarkModeUI[892] inverts luminance of colors in user interface of this component.

New global functional variable: RVGetSystem Color[1125] allows using custom color schemes in VCL applications.

Images

This version implements drawing TMetafile images using GDI+ function (VCL). It provides higher quality of drawing, including anti-aliased lines.

To use this feature, add RVGDIPlusGrIn unit in your project.

Equations

If the commercial version of Adit Math Engine is used, equation objects[193] can be saved and loaded in HTML and DocX files (as MathML and OMML objects, correspondingly). Saving can be controlled using properties of the object retuned by RVMathEquationManager[1054] function.

Files

Markdown

New optional parameter CodePage for TRichView.SaveMarkdown[348], SaveMarkdownToStream[349] allows to specify text encoding.

Universal loading methods

New optional CodePage parameter of LoadFromFile, LoadFromFileEx[328] and LoadFromStream, LoadFromStreamEx[329] specifies encoding for loading plain text and Markdown.

Database and LiveBindings

New option is added to TRVDBFieldFormat[1064]: *rvdbMarkdown*. It allows saving data to a database field in Markdown format.

New properties: TDBRichView[604].AllowMarkdown[608], TDBRichViewEdit[616].AllowMarkdown[621], TRichView[216].Document[238].AllowMarkdown[430] allow loading Markdown from database fields (instead of plain text)

New properties: TDBRichView[604].CodePage[609], TDBRichViewEdit[616].CodePage[622], TRichView[216].Document[238].CodePage[431] specify codepage for saving/loading Markdown and plain text from database fields.

2.3 New in version 22

Compatibility issues

- In FireMonkey, TRVGraphicFM (also accessible as TRVGraphic[1032]) is an abstract class now. Actual classes representing images are inherited from it (such as TRVRasterGraphicFM, also accessible as TRVBitmap)
- TRVGraphicClass[1033] declaration is changed for FireMonkey

RAD Studio 12 Athens

Delphi and C++Builder 12 are supported.

Skia

TRichView supports Skia4Delphi[163]. It is included in RAD Studio 12, or can be downloaded from ***https://skia4delphi.org/*** (version 6 is required).

In FireMonkey (add fmxRVSkiaFM unit in your project):

- support of Skia canvases
- PDF saving methods for TRVPrint and TRVReportHelper (see below)
- SVG image (see below)

In VCL (add RVSkia unit in your project):

- SVG image (TSkSvgGraphic is registered as SVG class for RVGraphicHandler[1125])

FireMonkey

New platforms:

- **Linux 64-bit**. Requirements: Delphi 10.3 and newer, FMXLinux 1.74 or newer.
- **iOS 64-bit**. Requirements: Delphi 10.4 and newer.
- **iOS Simulator ARM 64-bit**. Requirements: Delphi 11.2 and newer.

Files

DocX and RTF

- Since this version, TRichView can load DocX files and streams even in old versions of Delphi, see requirements.
- New event OnReadField[393] allows custom processing of RTF and DocX fields.
- Reading and writing of footnote and endnote numbering types in RTF and DocX. Reading is disabled by default, assign RTFReadProperties[253].ReadNoteNumberingType[465] = *True* to enable.

HTML

- Saving: new option for TRichView.HTMLSaveProperties[244].ImageOptions[442]: *rvhtmlsioImageSizeAttributes.*
- Loading: new property TRichView.HTMLReadProperties[243].XHTMLCheck[436] turns XHTML stricter syntax checking on/off.

PDF

- [FireMonkey with Skia4Delphi 163] New methods: TRVPrint.SavePDF 715 and TRVReportHelper.SavePDF 727.

Graphics

- SVG images are supported using Skia4Delphi. FireMonkey: TRVSvgImageSkiaFM 1032 class (include fmxRVSkiaFM unit in your project); VCL: TSkSvgGraphic (include RVSkia unit in your project).
- New formats are added in TRVGraphicType 1077: JPEG XR and WebP. RVGraphicHandler 1125 .LoadFromStream and GetDataGraphicType 1033 can detect these formats.
- [FireMonkey] RVGraphicHandler 1125.GetFileDialogFilter 1033 returns a file filter string that includes both bitmap and non-bitmap (such as SVG) formats that can be loaded.

Other

Change: Table.DeleteEmptyCols 929 and DeleteEmptyRows 929 keep selection in the table.

New tab alignment 856: decimal align to align floating point numbers in columns.

New property for images: *rvepSmoothScaling* 1067, it allows to turn smooth scaling on/off.

2.4 New in version 21

This topic included changes introduced in v21.1.

Compatibility issues

TRichViewEdit.SetItemTextEd 574 (and related methods) calls OnItemTextEdit 588 event.

TRVStyle.SaveCSS 772 and SaveCSSToStream 772 parameters changed (*rvcssUTF8* is removed from AOptions, new Encoding parameter is added).

TRichView.SavePicture 349 has a new parameters: InlineImage, IsBackgroundImage.

For macOS, **Ctrl** was changed to **Command** in default shortcuts and for hyperlinks activation in editor.

Deprecated methods and classes

HTML import: TrvHtmlImporter and TrvHtmlViewImporter classes are deprecated: use the new HTML loading and insertion methods.
HTML export: SaveHTMLEx 345, SaveHTMLToStreamEx 348, old versions of SaveHTML 343, SaveHTMLToStream 347 are deprecated. Use the new HTML saving methods.

HTML import (new)

Starting from this update, TRichView can load, insert and paste HTML.

New property: TRichView.HTMLReadProperties 243.

New methods:

- TRichView.LoadHTML 330, LoadHTMLFromStream 332;
- TRichViewEdit.InsertHTMLFromFileEd 529, InsertHTMLFromStreamEd 530;

- TRichViewEdit.PasteHTML[545], CanPasteHTML[507].

New event: TRichView.OnImportFile[385].

LoadFromFile[328], LoadFromStream[329], Paste[543] methods support HTML.

TDBRichView[604] and TDBRichViewEdit[616] components support HTML.

HTML export

Old HTML saving methods are deprecated.

New property: TRichView.HTMLSaveProperties[244].

New methods: SaveHTML[342] and SaveHTMLToStream[346]. They use properties from HTMLSaveProperties[244] and DocParameters[237].

Ability to save images inline inside HTML instead of external files.

HTML export is updated to use more CSS features.

Markdown

Markdown import and export support images stored inline inside Markdown text instead of external files.

New option in TRichView.MarkdownProperties[245].SaveOptions[456]: *rvmdsoInlineImages*.

New methods for loading and saving

LoadFromFile/LoadFromFileEx[328] load a file with format detection.

Design time

An update checker: a design-time tool that can check if a newer version of TRichView is available. It works when IDE starts (if allowed), or when the user choose "Check for Update" in components' context menu.

LiveBindings (DXE2+)

New TRichView property: Document[238]. This property can be linked to a DB field using LiveBindings. It contains sub-properties defining properties of this link.

The events OnNewDocument[391], OnLoadDocument[391], OnLoadCustomFormat[390], OnSaveCustomFormat[595] were moved from data-aware controls to their parent classes (because they are used for LiveBindings as well).

FireMonkey

Android 32-bit and 64-bit platforms are supported (in Delphi 10.4 and newer).

If PopupMenu property is not assigned,TRichView and TRichViewEdit controls display a standard popup menu.

Live spelling check[138] can be made both using third-party spelling checkers, and using a platform spelling check service (if available, see CheckSpelling[480] property). Suggestions for corrections can be shown in a popup menu (automatically for a standard menu, or using AddSpellingMenuItems[497]

for a custom menu). Suggestions are requested either from a platform spelling check service, or using OnGetSpellingSuggestions [382] event.

"SmartPopup" button can display not only a menu, but TPopup component (Popup [601] property).

Almost all FireMonkey demo projects were redesigned to fit screens of mobile devices.

For macOS, **Ctrl** was changed to **Command** in default shortcuts and for hyperlinks activation in editor (see rvoCtrlJumps in TRichViewEdit.EditorOptions [483])

Tables

New IgnoreContentHeight [958] property of table cells.

Text search without selection

The new method TRichView.StoreSearchResult [375] allows storing the position of the last found string from OnTextFound [418] event. This position can be used as a staring point for the next call of TRichView.SearchText [354]/TRichViewEdit.SearchText [553], if *rvsroFromStored/rvseoFromStored* is included SrchOptions parameter.

Custom drawing

New event TRichViewEdit.OnDrawCurrentLine [584] allows to highlight the current line.

Other

New optional parameters (ImageWidth and ImageHeight) in methods: TRichView.AddPicture [278] and AddHotPicture [274], TRichViewEdit.InsertPicture [534] and InsertHotPicture [528].

2.5 New in version 20

Compatibility issues

To use equation objects [193] with the free version of Adit Math engine, RVBasicMathWrapper must be included in project.

Some functions and constants were moved in another units (RVCoordFuncs, RVColorFuncs, RVClipboard).

When TRVStyle [739].MainRVStyle [756] is assigned, not only StyleTemplates [762], but all properties of TRVStyle are shared (except for collections of text, paragraph, and list styles).

RAD Studio 11 Alexandria

TRichView supports Delphi and C++Builder 11.

FireMonkey

TRichView [216], TRichViewEdit [469], TRVStyle [739], TRVPrint [696], TRVReportHelper [721] components are ported to FireMonkey.

Supported platforms:

- Windows 32-bit and 64-bit (RAD Studio XE6 and newer)
- macOS 64-bit (Delphi 10.3 and newer)
- macOS ARM 64-bit (Delphi 11 and newer)

See information about differences between VCL and FireMonkey versions [161].

Markdown

New methods for loading ***Markdown*** in TRichView: LoadMarkdown [332], LoadMarkdownFromStream [333], and inserting in TRichViewEdit: InsertMarkdownFromFileEd [532], InsertMarkdownFromStreamEd [532].

For LoadFromStream [329], you can choose loading either a plain text or a Markdown (a new optional parameter is added).

Details are explained in the topic about TRVMarkdownProperties [447] class.

RTF and DocX

New properties controlling RTF and DocX import:

- CanReuseNumberedLists [459]
- ConvertSymbolFonts [461]

Equation document objects

Both the free and the commercial version of Addit math engine are supported in TRichView equation objects [193].

To use the free version (included in TRichView setup), add RVBasicMathWrapper unit in your project.

To use the commercial version, add RVAdvMathWrapper unit in your project.

Custom drawing

New events for drawing additional content on top of images:

- TCustomRichView.OnAfterDrawImage [378]
- TCustomRVPrint.OnAfterPrintImage [885]

Selection drawing

A new mode of drawing selection is introduced: semitransparent selection. In this mode, instead of using special colors for drawing background and text of a selected fragment, this fragment is shaded with a special color.

New property: TRVStyle.SelOpacity [751].

For FireMonkey, there is an Option [247] to get selection color and opacity from the linked visual style: *rvoDefaultSelectionFill.*

Tables

New table operations:

- TRVTableItemInfo.MoveRows [938].

Other new features

- new list counter types: lower Greek and decimal-leading-zero for paragraph numbering [864].
- new global variable RichViewCtrlUpDownScroll [1112], specifies how Ctrl + Up and Ctr + Down keys are processed.
- new TRichViewEdit.MoveCaret [542] method.
- HTML saving improvement: for paragraph numbering, values of auto-calculated counters are saved only if necessary
- HTML from RTF extraction [1049]

2.6 New in version 19

This topic includes changes introduced in version 19.1.

Compatibility issues

- In TDBRichViewEdit [616], unused styles are deleted not only if AutoDeleteUnusedStyles [621] = *True*, but also if FieldFormat [623] = *rvdbRTF or rvdbDocX*.
- TRichView [216].SavePicture [349] parameters are changed.
- TRichView.RTFReadProperties.ErrorCode is renamed to RTFErrorCode [466].
- Backspace in tables [179] deletes the selected rows/columns (or the table itself, if all cells are selected). To turn off this feature, assign *False* to RichViewTableAutoAddRow [1117].

RAD Studio 10.4 Sydney

TRichView supports Delphi and C++Builder 10.4, including per-control VCL styling.

DocX (Microsoft Word Document) import

Since this version, the components can not only save but also load DocX files and streams. This feature requires Delphi 2009 or newer, or Lazarus.

New methods of TRichView: LoadDocX [326], LoadDocXFromStream [327].

New methods of TRichViewEdit: InsertDocXFromStreamEd [527], InsertDocXFromFileEd [526].

TRichView.LoadFromStream [329] can detect and load DocX.

TDBRichViewEdit [616] can store documents as DocX [623].

Markdown

New methods for exporting TRichView documents to ***Markdown*** files or streams: SaveMarkdown [348], SaveMarkdownToStream [349].

New property MarkdownProperties [245] contains sub-properties that specify Markdown saving options.

Other new features

New components:

TRVDataSourceLink[685] helps to use TRichView or ScaleRichView components as containers for data-aware controls.

New event of TRichView component:

- OnWriteObjectProperties[420] allows saving additional object properties in RTF, DocX, and HTML.
- OnTextFound[418] allows a special processing of found text (instead of / in addition to selecting)
- New optional parameter in TRVReportHelper[721].DrawPageAt[725].
- New property TRVReportHelper[721].TargetPixelsPerInch[723] allows overriding the target canvas's pixel depth.
- New optional parameter (SmoothScroll) in TRVScroller[869].ScrollTo[875].
- New option for text protection: *rvoFastDeleteProtectedText* option for TRichViewEdit.EditorOptions[483].
- High-quality image scaling is supported for transparent images too (for Delphi 2009+ and Lazarus)
- TRVReportHelper[721] uses high-quality image scaling, if AllowTransparentDrawing[723] and Preview parameter are True
- Backspace in tables[179] deletes the selected rows/columns (or the table itself, if all cells are selected).

Changes

In TRichViewEdit.OnItemResize[587] event, values of Val1 and Val2 parameters are changed.

TRichViewEdit.SetCheckpointInfoEd[560] and InsertCheckpoint[525] were changed from procedures to functions.

2.7 New in version 18

This topic includes changes of v17.3 and newer.

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[269] instead of AddBreakEx, AddBreakTag, AddBreakExTag.

All String, TRVRawByteString, TRVAnsiString properties and parameters of methods and events are changed to TRVUnicodeString[1099], where it is possible.

Changed events:

- In TRichView: OnAssignImageFileName[379], OnImportPicture[386], OnItemAction[387], OnItemHint[388], OnReadHyperlink[395], OnRVDblClick[399], OnRVRightClick[404], OnRVFControlNeeded[400], OnRVFPictureNeeded[401], OnSaveComponentToFile[405], OnSaveDocXExtra[407], OnSaveHTMLExtra[409], OnSaveImage2[410], OnSaveItemToFile[411] (additionally, Unicode: Boolean parameter is removed), OnSaveParaToHTML[413], OnSaveRTFExtra[414], OnSpellingCheck[416], OnWriteHyperlink[418]

- In TRichViewEdit: OnDropFile [586], OnItemTextEdit [588]
- In TRVStyle: OnDrawStyleText [781] (see also below)

A new PSaD parameter is added to custom drawing events of TRVStyle component. Additionally, since this version, text is drawn relative to its base line. These change may affect your existing custom drawing code. Also, parameters are added to:

- In TRVStyle: OnDrawStyleText [781] (Baseline and PSaD), OnApplyStyle [774] (PSaD)
- In TFontInfo: Draw [825] (Baseline and PSaD)

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.

In OnSaveDocXExtra [407], with Area = *rv_docxs_ParaStyle*, the meaning of Obj parameter is changed (now it contains an RVData instead of a paragraph style).

Many var-parameters of methods were changed to out-parameters. This change should not affect your projects unless you developed classes inherited from TRichView classes.

Controls loaded from RVF are now scaled from RVF PPI ("pixels per inch") to the actual PPI. Controls in RVF files saved by older versions of TRichView are not scaled (unless you change value of DefaultRVFPixelsPerInch [1109]), however, the actual PPI is assigned to their Font.PixelsPerInch.

RichViewPixelsPerInch [1115] global variable and TRVStyle.TextStyles.PixelsPerInch [786] are deprecated. Use TRichView.DocumentPixelsPerInch [239] and TRVStyle.UnitsPixelsPerInch [765].

TRVStyle.GetAsPixels [768], TRVStyle.PixelsToUnits [769], and functions for unit conversions [1044] have additional parameters.

Sizes measured in pixels now use TRVStyle.UnitsPixelsPerInch [765] instead of Screen.PixelsPerInch. This change affects some code, e.g. the example in RV_GetPrinterDC [1050];

Lazarus

TRichView is compatible with Lazarus [160] (32-bit and 64-bit Windows targets)

RAD Studio 10.3 Rio

TRichView is compatible with Delphi and C++Builder 10.3 Rio.

Complete support for multiple monitors having different PPI ("per monitor v2").

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 [824]
- TRVStyle.DefUnicodeStyle [746]
- TRVRTFReaderProperties.UnicodeMode [467]

New features related to Unicode:

- support for Unicode characters having codes greater than $FFFF
- support for pasting and drag&dropping of Unicode URLs

- 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); this feature can be turned off, if you include *rvoAlt0CodesUseKeyboardCodepage* in EditorOptions [483]
- entering a character by typing its hexadecimal code (directly in the editor) and then pressing **Alt** + **X**. Pressing **Alt** + **X** again converts it back to a hexadecimal number.

Measure units, zooming, high DPI

New measure unit is added to TRVStyleUnits [1094]: EMU (English metric unit).

You can define DPI for all values measured in pixels in TRVStyle.UnitsPixelsPerInch [765]. With this property, pixels become independent of the screen DPI, like twips and EMU. When displaying on the screen, all pixel sizes are scaled from UnitsPixelsPerInch [765], including image sizes (the only exceptions are controls and images from image lists).

High DPI display modes are supported completely.

If application supports monitors having different DPI (in Delphi 10.1+), TRichView uses DPI of its monitor. "Per monitor v2" is supported in Delphi 10.3.

You can change document DPI using TRichView.DocumentPixelsPerInch [239] property. This property allows to implement zooming.

Editing

A new (alternative) way to group editing operations for undo: BeginUndoGroup2..EndUndoGroup2 [505] (or BeginUndoCustomGroup2..EndUndoCustomGroup2 [505]). These methods allow grouping operations in several table cells.

New events

TRichView.OnReadMergeField [398] allows reading merge fields from RTF.

Text drawing

All text is drawn relative to its base line. Previously, by default, it was drawn relative to its top left corner. New drawing produces better result for scaled text (in TRVPrintPreview [635] or in ScaleRichView). This change affected TRVStyle.OnDrawStyleText [781] event and TFontInfo.Draw [825] method, see the compatibility issues above.

Saving and loading

RVF

- 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 [254]: *rvfoIgnoreGraphicClasses*, it allows ignoring names of graphic classes specified in RVF.
- "pixels per inch" value is now stored in RVF files; this value is used to scale loaded controls [174] from RVF PPI to the actual PPI; you can disable scaling by assigning 0 to the additional integer property [1119] *rveipcDPIScalable* (but the actual PPI is still assigned to Font.PixelsPerInch of controls).

DocX

- OnSaveDocXExtra [407], with Area = *rv_docxs_ParaStyle*, now you know the location of this paragraph in the document.

Animation

If an image contains frames of an animation, it can be played [124]. Previously, this kind of animation was implemented only for TBitmap pictures. Now it is available for any image type.

This feature is available if RVGridAnimate unit is included in the project.

Other features

- GetCurrentCharacter [1045] may return more than one characters, if they assemble a single glyph.
- some improvements in table layout algorithm.

2.8 New in version 17

This topic includes changes of v17.2.

Compatibility issues

- New parameter in TRVPageCountItemInfo [1011].CreateEx [1012]
- 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 [123], the editor gives a higher priority to URL than it was before
- *rvddHTML* is included in the default value of TRichViewEdit [469].AcceptPasteFormats [478] 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 [253].StoreImagesFileNames property is removed.
- Previously, the functions for enumerations of notes and text boxes [1052] returned hidden [106] notes/boxes if *rvoShowHiddenText* was included in Options [247], even if AllowHidden parameter was *False*. Now, AllowHidden parameter is processed strictly.
- TCustomRichView.OnURLNeeded event is removed (use OnWriteHyperlink [418])
- TRichView.RTFReadProperties.SetHeader and SetFooter are deprecated. Use TRichView.SetHeader [363] and SetFooter [362] instead. Note: they affect not only RTF and DocX, but RVF as well.
- type of the Sender parameter in TRVReportHelper.OnDrawHyperlink [886] and OnDrawCheckpoint [885] is changed from TRVReportHelper to TCustomRVPrint.

Mathematical expressions [193]

TRichView includes a new item type displaying mathematical formulas (TRVMathItemInfo [974]). It requires Delphi XE4 or newer.

Tables [901]

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

- HeadingRowColor [923], LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor;
- RowBandSize, ColBandSize [913].

A new optional parameter (EditCellAfter) is added in DeleteRows [929] and DeleteCols [928] methods.
New property: CellOverrideColor [912].
New event: OnDrawBackground [949].

RVF [129]

New DocObjects [235] property allows saving custom objects in RVF.

Headers and footers

TRichView,SetHeader [363] and SetFooter [362] define documents as headers/footers. They can be used not only for RTF (as before), but also for RVF and HTML.
A new option [1086] is available for HTML saving, *rvsoHeaderFooter*.

Printing

New property TRVPrint [696].VirtualPrinter [707] allows drawing pages on any canvas using the specified page size and DPI. New methods DrawPage and DrawPageAt [710].

New optional parameters in TRVPrint [696].Print [713] and PrintPages [714] allow printing all/odd/even pages, in normal/reverse order.

"Widow and orphan control" is implemented, see *rvpaoWidowOrphanControl* [833] option of paragraphs.

The following events and properties are moved from TRVReportHelper [721] to its parent class: OnDrawCheckpoint [885], OnDrawHyperlink [886], NoMetafiles [881]. Now you can use them in TRVPrint [696] as well.

A new property is added in TRVPrint [696] and TRVReportHelper [721]: MetafilePixelsPerInch [880].

TRVPageCountItemInfo [1011] now has NumberType [1009] property, like TRVPageNumberItemInfo [1010],

OLE Drag&drop [123]

New events of TRichViewEdit [469]: OnBeforeOleDrop and OnAfterOleDrop [580].

New option *rvoDragDropPicturesFromLinks* for TRichViewEdit.EditorOptions [483]. 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 [810]. 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[1114] global variable.

Loading external images

A new option is added to TRichView.Options[247]: *rvoAssignImageFileNames*. If included, the component assigns "image file name" properties when reading external images. Previously, RTFReadProperties[253].StoreImagesFileNames was used for this purpose for external images in RTF; this property is removed.

New event: OnAssignImageFileName[379] allows modifying file names before assigning to items/cells.

Item properties

rveipcPercentWidth additional integer property[1119] is added to controls[174].

rvepSpacing, rvepBorderWidth, rvepBorderColor, rvepOuterHSpacing, rvepOuterVSpacing integer properties[1067] work for labels[181].

rveipcRemoveSpaceBelow additional integer property[1119] is added to labels[181].

Caret

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

New global variables controlling the caret size and position[1108] are added. The new options include:

- setting the caret width equal to the next character/item width;
- using OnMeasureCustomCaret[588] event for a standard caret (not only for a custom caret);
- options for aligning wide carets horizontally.

Other changes

New TRichView.UseVCLThemes[264] property allows using colors of active VCL theme (in Delphi XE2+)

New TRichViewEdit event: OnDropFile[586]. It's more easy to use then OnDropFiles[586], and occurs not only on drag&drop, but also on pasting.

New TRichView event: OnCMHintShow[380] allows displaying custom hints.

Text flow options (ClearLeft[232] and ClearRight[232]) are shown as vertical lines around paragraph/line-break marks, if *rvoShowSpecialCharacters* is included in TRichView.Options[247]. Text flow options can be exported to DocX.

New method TRVStyleTemplateCollection.InsertFromRVST[801], it allows applying styles from a file to existing documents.

Packages are separated into runtime and designtime packages.

2.9 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 bidirectional 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 [829] = *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 [781]; you can return the old justification method by activating RVOLDJUSTIFY in RV_Defs.inc; however, this directive disables ParaStyle.Alignment [829] = *rvaDistribute* (it works like *rvaJustify*)
- TRVStyle.OnDrawStyleText [781] has new parameters.
- RichViewSafeFormatting constant is removed (replaced with RichViewStringFormatMethod [1116])
- 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 [763] property.

By default, Uniscribe text output is not compatible with metafiles. You can turn on metafile compatibility for TRVReportHelper [721] and TRVPrint [696] components using MetafileCompatibility [880] property.

New paragraph properties

New value for Alignment [829] – *rvaDistribute*. It aligns paragraph content to the both left and right sides by adding space between all characters.

New property LastLineAlignment [829] 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 [809]

- ParaStyle.Alignment [829] = *rvaJustify* (and *rvaDistribute*)
- TRVPrint.PreviewCorrection [882]
- 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 [959] are implemented for vertical text in table cells.

In addition to rotation [961] 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 [254]: *rvfoSaveStyleTemplatesOnlyNames*. It allows creating multiple RVF documents using the same collection of StyleTemplates [762].

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 [723] allows using methods for semitransparent drawing. It is useful for drawing over other images.

See the topic about semitransparent objects [126].

Support of high DPI screen modes

Size of touch screen selection handles [757] 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[551] deletes a page break from the current paragraph (not from the current item as before)

New RichViewSaveHyperlinksInDocXAsFields[1116] global variable.

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

2.10 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[1033])
  - (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[712], GetFooterRect[712].
- New values are added to TRVUndoType[1098].

New types of items

- sidenotes[186] and text box items[188] are printed in floating boxes.
- "page number" fields[190]
- "page count" fields[192]

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[216]: SaveDocX[341], SaveDocXToStream[342].

New OnSaveDocXExtra[407] event.

New options for TCustomRVFontInfo[806].Options[814]: *rvteoDocXCode* and *rvteoDocXInRunCode*.

New properties of TRVOfficeConverter[687]: ExcludeDocXImportConverter, ExcludeDocXExportConverter[690].

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[635] supports a two finger zooming as well.

Selection handles are implemented in TRichView[216] and TRichViewEdit[469]. 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[260] property.

New property: TRVStyle.SelectionHandleKind[757].

Printing and drawing

New properties defining sizes in TRVPrint[696]: Margins[705], HeaderY[705], FooterY[704], Units[707]. 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[716] and SetFooter[715] have additional optional parameter, specifying header/footer type. New properties: TitlePage[706], FacingPages[703].

New property: PageNoFromNumber[881] specifies the starting value for "page number" fields[190]. In TRVReportHelper[721], page numbers can be requested in OnGetPageNumber[728] event.

New TRVReportHelper[721].NoMetafiles property allows drawing metafiles as bitmaps.

Editing

New methods allowing to change properties of TRichViewEdit as an editing (undoable) operation: SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd[555], SetBackgroundImageEd[556]. They allow changing margins, DocParameters, background properties.

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

- SetCurrentItemExtraIntPropertyEx[566], SetItemExtraIntPropertyExEd[566];
- SetCurrentItemExtraStrPropertyEx[567], SetItemExtraStrPropertyExEd[567].

They support more properties than the old methods.

Text attributes

New property SizeDouble[817] allows defining font size with higher precision.

Visual changes

- New TRVStyle[739].FontQuality[748] property.
- Smooth image scaling[125].

Other changes

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

Footnotes[185]/endnotes[184] displaying a character (instead of a number) are supported.

All work with graphic classes is encapsulated in TRVGraphicHandler[1033] class, available via RVGraphicHandler[1125] variable.

RichViewJustifyBeforeLineBreak[1114] variable allow to turn on justifying lines before line breaks.

Faster processing of hidden text.

New properties:

- TRVScroller[869].NoHScroll[873];
- TCustomRichViewEdit[469].ForceFieldHighlight[486].

New option for TCustomRichView[216].Options[247]: *rvoPercentEncodedURL*.

2.11 New in version 14

Compatibility issues

- Item tags[96] 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[395], OnRVControlNeeded[400]). 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[401] 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[338], LoadTextFromStream[339], SaveText[352], SaveTextToStream[353], TRichViewEdit.InsertTextFromFile[541] 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[354] and TRichViewEdit.SearchText[553]). GetRVSearchOptions[1049] and GetRVESearchOptions[1049] 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[254], TDBRichView[604] clears BackgroundBitmap[229] before loading data from its field. If *rvfoLoadBack* is in RVFOptions[254] and FieldFormat[623]=*rvdbRVF*, TDBRichViewEdit[616] clears BackgroundBitmap[229] before loading data from its field.
- PNG images are saved to RTF as PNG images only if *rvrtfSavePngAsPng* is included in RTFOptions[252]. It is included by default. However, for existing projects, you may need to set it manually.
- The declaration of TRVLongOperation[392] type is changed.

- The following global variables were removed from RVTable.pas: RichViewTableGridStyle, RichViewTableGridStyle2, RichViewTableGridColor. They were replaced with properties of TRVStyle [750].
- A new option *rvoShowGridLines* in TRichView.Options [247] 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 [762] represent "real" text and paragraph styles. When style templates are used, an appearance of text and paragraphs are still defined in TextStyles [764] and ParaStyles [757]. 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 [263]=*True*.

TRichView.StyleTemplateInsertMode [260] specifies how style templates affect insertion of RVF and RTF data.

When the collection of style templates has been modified, OnStyleTemplatesChange [417] occurs.

In TRichViewEdit, you can apply the specified style template to the selection using the methods: ApplyStyleTemplate [501], ApplyTextStyleTemplate [502], ApplyParaStyleTemplate [500]. Style templates can be edited using ChangeStyleTemplates [508] method.

The function RVGetNoteTextStyleNo [1052] has an additional optional parameter: a name of a style template for using for a note character.

TRVStyle.MainRVStyle [756] property allows using a single collection of style templates for several documents.

See the overview topic about style templates [103].

Rotation

Table cells can be rotated [961] by 90°, 180°, or 270°.

A new method returning item coordinates: GetItemCoordsEx [307]; it takes cell rotation into account.

String tags

Since version 13.2, item tags [96] 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 [963].

64-bit

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

Note: TRVOfficeConverter [687] 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[635] adjusts its colors (background, page border, shadow) according to the current style.

Text files

A new optional parameter for TRichView.LoadText[338], LoadTextFromStream[339], SaveText[352], SaveTextToStream[353], TRichViewEdit.InsertTextFromFile[541]: 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*[176] and *hotspots*[178] is saved to RVF.

The parameters of TRichView.OnRVFPictureNeeded[401] were changed. Now, this event is called not only for pictures[169] and hot-pictures[172], but also for tables[179] 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[252]: *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[395]/OnWriteHyperlink[418] is not assigned.

HTML saving

TRichView.SaveHTMLEx[345] can save stretched background images (previously, they were saved as centered).

TRichView.SaveHTMLEx[345] saves list markers better (both in normal and rvsoMarkersAsText[1086] modes).

New "hidden" parameters in TRichView.OnSaveImage2[410] event.

Default processing if OnWriteHyperlink[418] is not assigned.

SaveHTMLEx[345] 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[762] property.
- All adjacent hypertext items[97] having the same non-empty tags[96] are highlighted below the mouse pointer; so visually they represent a single hyperlink.
- New properties[1067] for non-text items: background color, border width and color, spacing around the border.
- Footnote[185] and endnote[184] 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[549].

New property: AcceptPasteFormats[478] allows to restrict a list of formats that the editor can paste.

New property: DefaultPictureVAlign[483] defines an alignment for pasted and dropped images.

DB components

If *rvfoLoadBack* is in RVFOptions[254], TDBRichView[604] clears BackgroundBitmap[229] before loading data from its field.

If *rvfoLoadBack* is in RVFOptions[254] and FieldFormat[623]=*rvdbRVF*, TDBRichViewEdit[616] clears BackgroundBitmap[229] before loading data from its field.

Printing

Tables

New property for a table row: KeepTogether[967]. New table methods: SetRowPageBreakBefore, SetRowKeepTogether[945].

New option for table.PrintOptions[919]: *rvtoContinue*.

Other

New property: TCustomRVPrint.IgnorePageBreaks[879].

Tables

New properties of TRVStyle defining the appearance of grid lines: GridColor, GridStyle, GridReadOnlyColor, GridReadOnlyStyle[750]. They replace global variables used in older versions. New option *rvoShowGridLines* in TRichView.Options[247].

Other changes

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

New event of TRichViewEdit[469]: OnCheckStickingItems[583] event allows to stick items together.

New typed constant: RichViewDoNotMergeNumbering[1113].

"Smart start" search option for TRichView.SearchText[354] and TRichViewEdit.SearchText[553].

TRichView.OnProgress[392] 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[694] and ExportRV[693].

Better table layout algorithm.

New methods for TRVStyle: FindTextStyle, FindParaStyle[771], TextStyles.FindSuchStyleEx[791], ParaStyles.FindSuchStyleEx[794].

2.12 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[753] property (*rvWrapSimple* value). But even with this value, a line breaking can be different for Unicode[135] text in paragraphs with Alignment[829]=*rvaJustify*.
   RichViewWrapAnywhere (global constant) was removed, superseded by TRVStyle.LineWrapMode[753] (*rvWrapAnywhere* value).
2. New parameter in ApplyStyleConversion[501] and ApplyParaStyleConversion[499]. 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[1094] instead of bytes. All applications that use GetBreakInfo[296] and GetCurrentBreakInfo[512] 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[753] allows changing a line wrapping mode.

Hidden items

Some items can be hidden[106].

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[739].Units[764] property. Instead of pixels, you can use twips[144].

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[1115]=0.

Shared item content

A new item property is added: rvepShared[1067]. It allows for several picture items[169] to share the same graphic object. For a control item[174], it allows to reuse its control after the item deletion.

Incremental printing

TRVPrint[696] and TRVReportHelper[721] components have new properties: MinPrintedItemNo[881] and MaxPrintedItemNo[879]. 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[482] 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[501] and ApplyParaStyleConversion[499] have the new parameter (Recursive) allowing to prevent processing of table cells.

Drawing and formatting

Improved table layout algorithm.

New option in RVVisibleSpecialCharacters[1129]: *rvscTab*. RVParagraphMarks[1126] global variable allows defining how paragraph breaks and line breaks are displayed in "show special characters" mode.

New property TRVStyle.DisabledFontColor[746] allows change text color in disabled controls.

Calls of TRVStyle.OnDrawTextBack[783] 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[883] returns the page index for the given position in the document.

Saving and loading

Improved saving of footnotes[185] and endnotes[184] in plain text files.

Smarter replacing space characters to   when saving HTML.

New text properties for TRichView.DocParameters[237]:
- Author[424],
- Title[427],
- Comments[424]

Hypertext

rvoNoReadOnlyJumps option from EditorOptions[483] disallows automatic switching to a hypertext mode when TRichViewEdit becomes read-only[488].

2.13 New in version 12

Compatibility issues

Assignment to PageBreaksBeforeItems[250] requires reformatting even if the given item starts a paragraph (because it clears text flow around side-aligned items).

OutlineLevel[835] 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[167]. While the new behavior (searching in multiple text items) requires specifying a new option in the parameter of SearchText methods explicitly, GetRVSearchOptions[1049] and GetRVESearchOptions[1049] functions always return this new option.

Left- and right-aligned objects

Two new values are available for TRVVAlign[1100] type: *rvvaLeft* and *rvvaRight*. They can be used for all non-text items, except for tables[179], *breaks*[173], tabs[168], list markers[180], footnotes[185] and endnotes[184].

New methods are available for accessing vertical alignment:

- TRichView.GetItemVAlign[311],
- TRichView.SetItemVAlign[369],
- TRichViewEdit.GetCurrentItemVAlign[520],
- TRichViewEdit.SetItemVAlignEd[575],
- TRichViewEdit.SetCurrentItemVAlign[575].

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

- TRichView.ClearLeft[232],
- TRichView.ClearRight[232],
- TRichViewEdit.ClearTextFlow[509].

Showing invisible characters

New options for RVVisibleSpecialCharacters[1129] allowing to display/hide lines for floating objects, paragraph directions, printing options and outline levels[835].

New colors can be defined in TRVStyle:

- FloatingLineColor[747],
- SpecialCharactersColor[762].

Headings

New property in paragraph style: OutlineLevel[835] defines headings.

Other

New option of text searching (*rvsroMultiItem* for TRichView.SearchText[354], *rvseoMultiItem* for TRichViewEdit.SearchText[553]) allows the searched string matching substrings of several text items.

New event in TRVStyle: OnAfterApplyStyle[774].

New event in TRichView: OnGetItemCursor [382]

New property in TRVLabelItemInfo: RemoveInternalLeading [972]

New methods in TRVOfficeConverter: IsValidImporter [695] allows checking a file before importing.

New values in TRichView.RVFOptions [254] and RVFWarnings [258] (*rvfoIgnoreUnknownCtrlProperties*, *rvfwUnknownCtrlProperties*)

2.14 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 [1056], TRVUnicodeString [1099], TRVRawByteString [1085]. 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 [1085]: OnDrawStyleText [781], OnReadHyperlink [395], OnItemAction [387], OnSaveItemToFile [411], OnRVDblClick [399], OnRVRightClick [404], OnItemTextEdit [588]. In existing projects, change types of parameters in these events manually.

In OnSaveComponentToFile [405] 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 [1085] 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 (277) (and its versions);
- GetWordAt (322);
- GetItemText (310);
- SetItemText (368);
- GetCurrentItemText (519);
- SetItemTextEd (574);
- SetCurrentItemText (568).

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

- CopyText (290);
- GetSelText (321);
- InsertStringTag (539);
- InsertText (540);
- PasteText (548);
- SearchText (354).

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

- AddFmt (273);
- AddItem (276);
- InsertItem (531);
- GetTextInfo (321);
- GetCurrentTextInfo (523).

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) (135);
- removing text formatting (1139).

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* strings (96) (in tags-are-PChars mode);
- extra item properties (1072);
- names of *checkpoints* (92);
- visible text in labels (181), numbered sequences (183), endnotes (184), footnotes (185);
- text in list markers (180) (not only bullets, but all numbering);
- live spelling (138) interface;
- hypertext targets (OnReadHyperlink (395), OnWriteHyperlink (395));
- and others.

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

- Unicode property of text style (from False to True);
- TRichView.RTFReadProperties.UnicodeMode (from *rvruNoUnicode* to *rvruOnlyUnicode*);
- TRichView.Options (*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 property of text style is saved. When saving text styles (in RVF files or Delphi forms) in Delphi/C++Builder 2009, value of Unicode 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, OnReadHyperlink, OnItemAction, OnSaveItemToFile, OnRVDblClick, OnRVRightClick, OnItemTextEdit) must be converted to String before using. For example, for OnRVDblClick event:

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

New methods, properties and events

New TCustomRichView.OnAddStyle event.

New NoVScroll property of TRichView.

New options for line spacing ("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.

New properties for RTFReadProperties: CharsetForUnicode and UseCharsetForUnicode.

RVF:

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.

2.15 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[345] 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[411] 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[1117] to *False* to disable.

Editing

- BeginUpdate[506] and EndUpdate[512] have new functionality.
- Simple version GetWordAt[322] always returns ANSI string.
- Paste[543] can paste graphic files in CF_HDROP format.

RTF

- bookmarks are read from RTF as checkpoints[92].
- Local links are converted automatically when reading and writing RTF.
- Sub/superscripts and VShift[820] are saved and loaded to RTF differently.

Other

- meaning of ItemNo parameter in OnControlAction[380] is changed.
- RVF insertion methods ignore background.

Main new features

- Animation of pictures[124];
- "Smart popups"[598];
- page breaks inside table cells.

New item types[90]

- labels[181];
- sequences[183];
- footnotes[185];
- endnotes[184];

- references to parent notes [190].

New in saving and loading

HTML:

New options in Options parameter of SaveHTML [343] and SaveHTMLEx [345]:

- rvsoXHTML [1086] – if included, XHTML is saved instead of HTML;
- rvsoUTF8 [1086] – if included, HTML/XHTML is saved in UTF-8 encoding.

You can choose between using <p> and [1107].

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 [237] property.

Options [252] for saving smaller RTF files: *rvrtfSavePicturesBinary*, *rvrtfPNGInsteadOfBitmap*.

RTF bookmarks are read as *checkpoints* [92]. 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 [253]:

- StoreImagesFileNames allows storing file names of external images (update: this property is removed in version 16);
- BasePathLinks [459] specifies whether to add path to file in hyperlinks;
- IgnoreBookmarks [463] allows ignoring bookmarks;
- IgnoreNotes [463] allows ignoring footnotes and endnotes;
- IgnoreSequences [463] allows ignoring numbered sequences;
- IgnoreTables [464] allows ignoring tables;
- ReadDocParameters [465] allows reading DocParameters;
- UseSingleCellPadding [468] sets mode for loading tables cell padding.

Text:

- You can allow saving explicit page breaks in text files/streams. Set RichViewSavePageBreaksInText [1107] typed constant to *True*.

Other:

- LoadFromStream [329] loads document in RVF, RTF or text formats (autodetect).

New in DB controls

- TDBRichView [604] and TDBRichViewEdit [616] can be placed in TDBCtrlGrid.
- TDBRichView [604] and TDBRichViewEdit [616] can update standard DB actions.
- TDBRichView [604].OnLoadCustomFormat [390], TDBRichViewEdit [616].OnLoadCustomFormat and OnSaveCustomFormat events allow loading and saving data to fields in custom formats.
- Special support for Unicode memo fields.

New in drag&drop [123]

Low level events:

- OnOleDragEnter[589];
- OnOleDragOver[590];
- OnOleDragLeave[589];
- OnOleDrop[591].

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[354] and TCustomRichViewEdit.SearchTextW[553] allows searching for WideString.

New in layout and formatting

- If you set RichViewShowGhostSpaces[1116] typed constant to *True*, spaces hidden between lines will be shown if *rvoShowSpecialCharacters* is in TCustomRichView.Options[247].
- You can define which characters to show in ShowSpecialCharacters[247] mode. They are listed in RVVisibleSpecialCharacters[1129] typed constant.
- New possible values for TCustomRichView.BackgroundStyle[229].
- New property TCustomRichView.MaxLength[246] – maximal count of characters per line.
- New property TCustomRichView.WordWrap[265] 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[62]).

New text attributes

- New text Protection[815] options: *rvprStyleSplitProtect*, *rvprSticking2*, *rvprSticking3*.
- New property HoverEffects[812] allows hyperlinks underlining.
- New properties UnderlineType[819], UnderlineColor[818] and HoverUnderlineColor[813] allow to customize underlines.
- New property SubSuperScriptType[818] allows implementing subscripts and superscripts.

New in printing

- TRVReportHelper[721] and TRVPrint[696] do not require parent control any more. They can be placed on data module or created without owner.
- TRVPrintPreview[635] displays full paper size, including areas where printer cannot print (see WinAPI function GetDeviceCaps, PHYSICAL*** constants).
- TRVPrintPreview.PrintableAreaPen[893] 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[703] property.

New in item properties

- *rvepVisible* item property[1067] allows to hide controls[174];
- *rvepResizable* item property[1067] works for pictures[169] too;
- *rvespAlt* item property[1072] can be applied to *bullets*[176] and *hotspots*[178] as well as to pictures[169] and *hot-pictures*[169];

- new values for VAlign[1100] property of items: *rvvaAbsTop*, *rvvaAbsBottom*, *rvvaAbsMiddle*.
- new styles[1059] for *breaks*[173] (horizontal lines): dashed and dotted; all types of *breaks* are drawn without rounded corners.

New in *checkpoints*[92]

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

- InsertCheckpoint[525];
- GetCheckpointAtCaret[512];
- RemoveCheckpointAtCaret[550].

New in tables[179]

- Table cells can be printed on several pages (previously, page breaks were possible only between rows).
- New option for table.Options[916]: *rvtoNoCellSelect*. If included, multicell selection by mouse is not allowed.
- New property table.VisibleBorders[921] allows to hide some sides of table border. Direct assignment to this property cannot be undone/redone, use table.SetTableVisibleBorders[946] 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[129] 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[1117] to *False* to disable this feature).
- New properties RowCount[920], ColCount[914].
- New properties CellHPadding[911], CellVPadding[913].
- New property Cell[951].Hint[958].
- New property Row[966].PageBreakBefore[967].
- New methods SetCellBackgroundImageFileName[941] and SetCellHint[943].
- Improved table layout algorithm.
- better SplitSelectedCellsVertically[946] and SplitSelectedCellsHorizontally[946] implementations.

New in RVF[129]

If image was not returned in OnRVFPictureNeeded[401] event, this event is called the second time with the value of *rvespImageFileName*[1072] 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[483], measured in OnMeasureCustomCaret[588] event, drawn in OnDrawCustomCaret[585] 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[544].
- New characters are included in the default value of TCustomRichView.Delimiters[235].
- meaning of TCustomRichViewEdit's BeginUpdate[506] and EndUpdate[512] was changed.
- New option for TCustomRichView.SearchText[354]: *rvsroFromStart*.
- Changes in TCustomRichView.GetWordAt[322].
- Ability to delete unused styles in multiple documents, see TRVDeleteUnusedStylesData[1031].
- Much faster reformatting, especially if *rvoFastFormatting* is included in TCustomRichView.Options[247] (default). Much faster loading RTF files. Much faster copying to the Clipboard.
- Function RVIsCustomURL[1126].
- 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[380].

2.16 New in version 1.9

Compatibility issues

- TRVStyle.CurrentItemColor[744] is used for drawing frame around the current item instead of TRVStyle.HoverColor[750].
- ApplyTextStyle[502], ApplyParaStyle[498], ApplyStyleConversion[501], ApplyParaStyleConversion[499] are applied to cells of all selected tables.
- OnSaveImage2[410] event: ImageSaveNo is declared as a var-parameter (it does not make sense otherwise)
- RTF export: only Standard[804] styles can be exported as style sheet.
- Default value of RVStyle.SpacesInTab[761] is changed from 8 to 0.
- Default value of TRichViewEdit.EditorOptions[483] is changed from [rvoClearTagOnStyleApp] to [rvoWantTabs, rvoCtrlJumps].
- New characters are added to default value of Delimiters[235] property.

New: tab stops

- Tabs[836] – properties of paragraph style defining tab stops.
- TRVStyle.DefTabWidth[745] defines default distance between tab stops.
- TRVStyle.SpacesInTab[761]: if equals to 0, methods for adding/inserting multiline text, RTF, and Tab key inserts tabulator[168] (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[280], TRichViewEdit.InsertTab[540] adds/inserts tabulator[168]

New: Live spelling check

See overview topic [138]

New in TRichView

- VAlign [264] property, ClientToDocument [286] method;
- *rvoShowSpecialCharacters* in Options [247];
- BeginOleDrag [284] – allows drag&drop of inserted controls;
- Reformat [339] method – formatting method that keeps selection;
- AddTextNLA [281] method;
- LoadTextFromStream, LoadTextFromStreamW [339];
- Events: OnItemHint [388], OnProgress [392].

New item properties

- additional string properties [1072]: hint, alt text, image file name; a set of methods supporting them: TRichView's GetItemExtraStrProperty [308] and SetItemExtraStrProperty [367], TRichViewEdit's GetCurrentItemExtraStrProperty [519], SetCurrentItemExtraStrProperty [567] and SetItemExtraStrPropertyEd [573] (this set of methods is similar to methods working with additional integer properties [1067]);
- *rveoNoHTMLImagesSize* additional integer property [1067]

Printing

Added: *rvpaoKeepLinesTogether* and *rvpaoKeepWithNext* Options [833] for paragraph style.

Tables

- *rvtoIgnoreContentWidth* and *rvtoIgnoreContentHeight* options for TRVTableItemInfo [901].Options [916];
- AssignProperties [925] method;
- better table layout algorithm.

RTF

- TRichView.RTFReadProperties [457].ExtractMetafileBitmaps [462] – 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 [345] is modified to conform "HTML 4.01 Transitional" standard, see http://validator.w3.org/;

- improvements in CSS export;
- SaveHTMLEx[345] saves characters of "Symbol" font as HTML entities (like α);
- better saving of RTF-like tables (with CellHSpacing[911]=CellVSpacing[913]=-1 to simulate 1-pixel width border) with SaveHTMLEx[345], or SaveHTML[343] with *rvsoForceNonTextCSS* in the Options parameter;
- ImagesPrefix parameter of SaveHTML[343]/SaveHTMLEx[345] can contain a full path (determined by the presence of ':').

TRVOfficeConverter[687]

- added: PreviewMode[691] property.

TRVReportHelper[721]

- added: OnDrawCheckpoint[885] event

Database controls

- OnLoadDocument event for TDBRichViewEdit.
- OnNewDocument and OnLoadDocument for TDBRichView.

Other improvements

- TRVStyle.CurrentItemColor[744].
- *rvoNoCaretHighlightJumps* in EditorOptions[483];
- faster redrawing on typing;
- better processing of bidirectional text[137];
- ApplyTextStyle[502], ApplyParaStyle[498], ApplyStyleConversion[501], ApplyParaStyleConversion[499] are applied to cells of all selected tables;
- all TRichViewEdit methods for inserting one item[119] 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($200B)) in Unicode text. Normally soft hyphen is invisible. It is displayed as a hyphen only if line breaks is after it.If *rvoShowSpecialCharacters* is in Options,[247] it is shown as "not" sign (WideChar($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[339] 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.

2.17 New in version 1.8

Compatibility issues

Changes that can affect existing code:

- OnHTMLSaveImage [383] is called for tables to save background image;
- OnHTMLSaveImage [383] 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 [246] property does not include left and right margins any more, so now it works according to the help file [110];
- parameters are changed in OnChanging [582] event (should not be a problem because this event is rarely used);
- table.OnDrawBorder [949] event has two new parameters: Row and Col;
- changes in *rvteoHTMLCode* and *rvteoRTFCode* (Options [814] 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 [758];
- RVF version number is changed from 1.2 to 1.3 (see RVF specification [148]); 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* [815] protection option meaning was changed (split into 3 rvprStick*** options).

Drag&Drop

See overview [123].

Background images for tables and cells

- table.BackgroundImage [904], BackgroundStyle [905];
- cell.BackgroundImage [953], BackgroundStyle [954].

Resizing pictures and controls with mouse

By default, all pictures [169] and hot-pictures [172] can be resized with the mouse (except for TIcon-s).

By default, controls [174] cannot be resized. But you can explicitly allow resizing for the given control, if you set its *rvepResizable* [1067] property to non-zero value. See also SelectControl [355] method.

You can disallow resizing in the given RichViewEdit by including *rvoNoImageResize* in its EditorOptions [483] property.

Extra items' integer properties [1067]

- *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[878] property for TRVPrint[696] and TRVReportHelper[721]: allows black and white or grayscale printing/previewing for color documents;
- much more accurate print preview[635];
- new properties of TRVPrintPreview: ClickMode[891], Color[891], PageBorderColor[893], PageBorderWidth[893], ShadowColor[894], ShadowWidth[894].

Selecting:

- new line selection mode, see TRVStyle.LineSelectCursor[753]
- defining visual appearance of selection, see TRVStyle.SelectionStyle[759]
- defining selection mode, see TRVStyle.SelectionMode[758]

RVF[129]

- 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[238]).

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[409] event now allows to insert HTML code before the </body> tag;
- new OnSaveParaToHTML[413], OnSaveImage2[410], OnSaveItemToFile[411] events;
- new options[1086] for HTML export;
- OnWriteHyperlink[418] event supersedes OnURLNeeded.

RTF export

- OnWriteHyperlink[418] event supersedes OnURLNeeded;
- new OnSaveItemToFile[411] 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[386]

DB controls

- new event: TDBRichViewEdit.OnNewDocument;
- new property: TDBRichViewEdit.IgnoreEscape [624].

Other improvements

- TRVReportHelper.OnDrawHyperlink [886] event;
- default text styles for paragraph styles (TParaInfo.DefStyleNo [839]);
- improvements in processing of bidirectional text [137];
- new action in OnItemAction [387] event: *rviaInserted*;
- faster redrawing on editing;
- previously reserved break styles [1059] are implemented;
- RefreshListMarkers [340] method.

2.18 New in version 1.7

Compatibility issues

- TCustomRichView.DeleteLines is renamed to DeleteItems [291].
- If you assign non-default values to extra item properties [1067], saved RVF files will not be opened with applications that use older version of RichView.
- OnDrawPageBreak [778] 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 [400] was changed; if you already use this event, you must change declaration of event handler manually;
- FirstIndent [831] is not applied to line starts inside paragraphs (Shift + Enter).
- DeleteUnusedStyles [293] has a new parameter (deleting list styles).
- GetWordAt [322] parameters are changed.
- Accessing invalid item in RVStyle.TextStyles and ParaStyles does not cause "List index is out of bounds" error any more.
- OnHTMLSaveImage [383] is called for background bitmap [229]; 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 [755]

Methods of TCustomRichView:

- SetListMarkerInfo[369],
- GetListMarkerInfo[315].

Methods of TCustomRichViewEdit:

- ApplyListStyle[498],
- RemoveLists[552],
- ChangeListLevels[508].

New options[1086] for HTML export – *rvsoMarkersAsText*.

New parameter in DeleteUnusedStyles[293] (deleting list styles)

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

Methods of TCustomRichView:

- SetItemExtraIntProperty[366] – changes value of property (without document reformatting);
- GetItemExtraIntProperty[307] – returns value of property.

Methods of TCustomRichViewEdit:

- SetItemExtraIntPropertyEd[572] – changes value of property, updates the document;
- SetCurrentItemExtraIntProperty[566] – the same for item at the position of caret;
- GetCurrentItemExtraIntProperty[518] – returns value of property of item at the position of caret.

Soft page breaks (preliminary implementation)

Methods of TCustomRichView:

- AssignSoftPageBreaks[284], ClearSoftPageBreaks[286] - displays/hides soft page breaks

Property of TRVStyle

- SoftPageBreakColor[761]

Other improvements in TCustomRichView

New methods:

- DeleteParas[293] – deletes the specified paragraphs, updates the document;
- GetLineNo[314] – returns line for the specified position;
- GetWordAt[322] (parameters are changed);
- GetItemAt[305] – returns item in the given coordinates.

New events:

- OnItemAction[387] – occurs on operations with item;
- OnPaint[392].

New properties:

- RTFReadProperties.ConvertHighlight[460], LineBreaksAsParagraphs[464].

New Options[1086] for HTML export: *rvsoUseCheckpointsNames.*

New Options[254] for RVF: *rvfoSaveLayout*, *rvfoLoadLayout*.

Events:

- OnSaveHTMLExtra[409], OnSaveRTFExtra[414] – allow saving additional information in HTML and RTF.

Other improvements in TCustomRichViewEdit

Methods:

- ApplyParaStyleConversion[499] (and OnParaStyleConversion[592] event) – implements commands like "change paragraph alignment", "increase indent", etc.
- InsertText, InsertTextW[540] have additional optional parameter;
- GetCurrentLineCol[521] – returns line and column at the caret position;
- ConvertToPicture[510], ConvertToHotPicture[510] – convert picture to *hot-picture* and vice versa

Properties:

- Modified[487]: Boolean

New EditorOption[483] – *rvoDoNotWantShiftReturns*.

Other improvements in TRVStyle[739]

- InvalidPicture[752] – picture that used instead of invalid or missed pictures;
- InvalidItem[786] for TextStyles, InvalidItem[792] for ParaStyles – are returned when accessing item which is out of index range.

New properties of text style (TFontInfo[821]):

- Options[814] – affects HTML and RTF export

New properties of paragraph style (TParaInfo[828]):

- new protection Option[833] – *rvpaoDoNotWantReturns*.

Improvements in TDBRichViewEdit[616]

- AutoDeleteUnusedStyles[621]

Improvements in TRVOfficeConverter[687]

- ExcludeHTMLImportConverter, ExcludeHTMLExportConverter[690];
- ExtensionsInFilter[690];
- ErrorCode[689] – returns error code for the last import/export operation.

Improvements in TRVPrint[696]

- OnPrintComponent[887] is optional since this version;
- ClipMargins[703] property – disallows printing on margins;
- headers and footers: TRVPrint.SetHeader[716], SetFooter[715], 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[363], SetFooter[362], DocParameters[237]).
- OnPagePostpaint[718];
- ColorMode[878] – allows black&white and grayscale printing and preview;

Improvements in all visual controls of the package

- UseXPThemes [873] - allows using WindowsXP themes (visual styles)

Improvements in tables [196]

Global variable RichViewTableGridStyle: TPenStyle – drawing invisible table borders.

Methods:

- SetCellVisibleBorders [945] – sets VisibleBorders property for the cell.

Properties:

- HeadingRowCount [914] – defines table header;
- new Option [916] – *rvtoRTFAllowAutofit*.

Events:

- OnCellEditing [948] – allows to prevent cell editing;
- OnDrawBorder [949] – custom drawing for cell borders.

New help topics

- Valid Documents [143]
- Custom Drawing [146]
- Examples for DBRichViewEdit [616]
- New overview: Item Types [164]
- 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.

2.19 New in version 1.6

Compatibility issues

- TRVPrint.OnPrintComponent [887] 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 [293].

Changes in TRVPrint

- new: TRVPrint.TransparentBackground [882]
- new: TRVPrint.GetFirstItemOnPage [883]
- new: TRVPrint.MirrorMargins [706]

Changes in tables

- Vertical alignment of cells (cells' VAlign 963 and rows' VAlign 968, SetCellVAlign 945 and SetRowVAlign 945 of table).
- BorderColor 956 and BorderLightColor 956 of cells allow to override default values (see also SetCellBorderColor 942 and SetCellBorderLightColor 943).
- Ability to hide table grid in editor (new option for table Options 916 – *rvtoHideGridLines*).
- Ability to print table background in white (new option for table PrintOptions 919 – *rvtoWhiteBackground*).
- New table Options 916 – *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 916). Previous versions show grid only for zero-width borders.

New components

TRVOfficeConverter 687 – allows using Microsoft Office converters

TRVReportHelper 721 – reserved for future use.

HTML export

New options 1086 – *rvsoImageSizes*, *rvsoForceNonTextCSS.*

RTF export

You can save hyperlinks in RTF, using OnURLNeeded event (update: this event is removed in v17; use OnWriteHyperlink 418).

New option 252 for saving RTF files – *rvrtfSaveEMFDefault*.

RTF import

- new: reading hyperlinks: OnReadHyperlink 395 event;
- reading tables (see also: RTFReadProperties.AutoHideTableGridLines 459);
- reading external images (see important note for using third-party graphics formats 1137);
- new: ignoring hidden text (see RTFReadProperties.SkipHiddenText 466).

Horizontal scrolling

new: HScrollPos 872, HScrollMax 871, OnHScrolled 877.

Visual effects and formatting

- new: TRVStyle.InactiveSelColor and InactiveSelTextColor 751;

- new: CharSpacing [809] property of text style.

DB Components

DB components can display RTF and plain text fields.

new: TDBRichViewEdit.FieldFormat [623] 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 [539].
- Parameters are added in TRichView.DeleteUnusedStyles [293].
- Essential styles can be marked with Standard [804] property.
- TRichView.SearchText [354] and TRichViewEdit.SearchText [553] now searches in table cells.
- Hypertext works automatically in read-only [488] editors (user does not need to press and hold Ctrl).
- Editor can hide caret in read-only mode (new option in EditorOptions [483] - *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 [554].

2.20 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 [36].

3. New parameter

New parameter of SaveCSS [772]/SaveCSSToStream [772]. Please use empty set ([]) for it.

New feature: RTF Import

- TRichView.LoadRTF [334];
- TRichView.LoadRTFFromStream [335];
- TRichViewEdit.InsertRTFFromFileEd [536];
- TRichViewEdit.InsertRTFFromStreamEd [536];
- TRichViewEdit.PasteRTF [547];
- TRichViewEdit.CanPasteRTF [507];
- and property TRichView.RTFReadProperties [253].

New feature: saving text and paragraph styles in RVF

- New *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions [254]
- TRichView.RVFTextStylesReadMode [258];
- TRichView.RVFParaStylesReadMode [258];
- Related: TRichView.DeleteUnusedStyles [293].

New feature: *hot pictures*

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

New methods: AddHotPicture [274], AddHotPictureTag [274], InsertHotPicture [528].

Old methods (GetPictureInfo[318], SetPictureInfo[370], GetCurrentPictureInfo[522], SetCurrentPictureInfo[569], SetPictureInfoEd[576]) now work with both pictures and *hot pictures*.

New properties for text and paragraph formatting:

- TFontInfo.StyleEx[818] (new option – *rvfsAllCaps*);
- TFontInfo.CharScale[808];
- TParaInfo.LineSpacing[832];
- TParaInfo.LineSpacingType[833];
- TParaInfo.Options[833].

New feature: Tab navigation through links and inserted controls

TRichView.TabNavigation[262] property;

TRichView.GetFocusedItem[302] 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[295].

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

Miscellaneous

- OnCaretMove[582]
- GetLastCheckpoint[313], GetPrevCheckpoint[319]
- two new possible values of Direction parameter of OnCaretGetOut[581]
- TRVStyle.ParaStyles.FindStyleWithAlignment[794] and FindSuchStyle[794]
- "whole word" option for TRichView.SearchText[354] and TRichViewEdit.SearchText[553] (thanks to Louis Kessler)

Improved:

- support of third party graphic classes[1136]
- RTF export
- HTML export: SaveHTML[343] now retains right and justify alignments.

2.21 New in version 1.4

Compatibility issues

- TRichView.OnControlAction[380] 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[195].

New methods

TRichView:

- AddNLATag[277], AddItem[276], GetItem[304], GetItemNo[309],
- GetJumpPointLocation[312]
- GetItemCoords[306], GetItemClientCoords[306]

TRichViewEdit:

- InsertItem[531],
- GetCurrentItem[517], GetCurrentItemEx[517]
- SetItemTextEd[574],SetCurrentItemText[568], GetCurrentItemText[519],
- CanChange[506]
- BeginItemModify[503], EndItemModify[511]
- ResizeCurrentControl[552], AdjustControlPlacement2[497]

TFontInfos:

- FindStyleWithCharset[788]

New properties

- TParaInfo.NextParaNo[840]
- TRVStyle.DefCodePage[745]
- TRichView.HScrollVisible[872]
- TRichView.RVData[254]
- TRichView.InplaceEditor[872]
- New option (rvoAutoSwitchLang) in TRichViewEdit.EditorOptions[483] (thanks to Pavel Zhelty)
- New states in TRVTextDrawStates[1097]

New events:

- TRichView.OnCopy [381]
- TRichView.OnHTMLSaveImage [383]
- TRichViewEdit.OnCaretGetOut [581]
- New "hidden" parameters in TRVStyle.OnDrawStyleText [781] and OnDrawTextBack [783]

Cursors

Two new cursors. Ability to use system default hypertext cursor [147]

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)

2.22 New in version 1.3

The main additions in version 1.3

- Undo/Redo [120] (many topics, this is the link to the overview);
- Unicode support [135] (many topics, this is the link to the overview);
- Events for custom drawing [773] (several topics; this is the link to the list)
- TRichView.RTFOptions [252], CopyRTF [290], SaveRTF [350], SaveRTFToStream [351]
- TRVPrint.OnPagePrepaint event
- TFontInfo.Protection [815] property
- TRVStyle.UseSound [766] property
- TRichView.AddTextBlockNL, SetItemText [368], GetItemText [310], SaveTextToStream, SaveTextToStreamW [352] methods;
- TRichView.OnControlAction [380] event;
- new options in TRichView.Options [247], TRichViewEdit.EditorOptions [483];
- new parameter in TRichView.LoadText [338];
- TFontInfo.IsEqual [821], TFontInfos.FindStyleWithFontStyle [790], FindStyleWithFontSize [790], FindStyleWithColor [788], FindStyleWithFontName [789], FindSuchStyle [791], FindStyleWithFont [789]
- new global variables [1108]
- 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[815].
- Older version of RichView will not be able to read RVF[129] with Unicode.

2.23 New in version 1.2

The main additions in version 1.2

- Borders and Background of Paragraphs[102];
- spacing between paragraphs (TParaInfo.SpaceBefore[836],TParaInfo.SpaceAfter[835]) and ability to move items to the new line inside paragraphs (Shift+Enter in editor, also see TRichView.IsFromNewLine[325], TRichView.IsParaStart[325], TRichView.SetAddParagraphMode[357]);
- ability to specify explicit page breaks (TRVStyle.PageBreakColor[756], TRichView.PageBreaksBeforeItems[250], TRichView.Options[247], TRichViewEdit.InsertPageBreak[534], TRichViewEdit.RemoveCurrentPageBreak[551]);
- ability to change background color of text under mouse (TFontInfo.HoverBackColor[811]);
- ability to save and load styles to registry (TRVStyle.LoadReg[771], TRVStyle.SaveReg[773], Delphi 4+ is required);
- ability to search upwards in RichView (TRichView.SearchText[354]);
- ability to save background in RVF file or stream (optionally) (TRichView.RVFOptions[254], RVF Overview[129]).

Compatibility issues

- default search direction in TRichView.SearchText[354] 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[772]);
- 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[811]

- TRVStyle.LoadReg [771]
- TRVStyle.SaveReg [773]

- TRVRect [1038]
- TRVBooleanRect [1018]
- TRVBorderStyle [1059]

- Borders and Background of Paragraphs [102]

- TParaInfo.SpaceBefore [836]
- TParaInfo.SpaceAfter [835]
- TParaInfo.Border [831]
- TParaInfo.Background [830]

- TRVBorder [852]
- TRVBorder Properties [853]
- TRVBorder.BorderOffsets [853]
- TRVBorder.Color [853]
- TRVBorder.InternalWidth [853]
- TRVBorder.Style [854]
- TRVBorder.VisibleBorders [854]
- TRVBorder.Width [854]

- TRVBackgroundRect [851]
- TRVBackgroundRect Properties [851]
- TRVBackgroundRect.BorderOffsets [851]
- TRVBackgroundRect.Color [852]

- TRichView.IsFromNewLine [325]
- TRichView.SetAddParagraphMode [357]

- TRVStyle.PageBreakColor [756]
- TRichView.PageBreaksBeforeItems [250]
- TRichViewEdit.InsertPageBreak [534]
- TRichViewEdit.RemoveCurrentPageBreak [551]

- TRVStyle.SaveCSSToStream [772]
- TRichView.SaveHTMLToStream [347]
- TRichView.SaveHTMLToStreamEx [348]

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

- RichView Paragraphs [100], Building RichView Document [107]
- Export of RichView contents to HTML [132], RichView Methods for Saving and Loading [127], Searching and Replacing [115],
- RVF Overview [129], RVF Specification [148]
- TRVSearchOption [1091]
- TRichView.SearchText [354]
- TRichViewEdit.SearchText [553]

- TRichView.IsParaStart 325
- TRichView.Options 247
- TRichView.RVFOptions 254
- TRichView.SaveHTML 343
- TRichView.SaveHTMLEx 345
- TRVSaveOption 1086
- TRichView.SavePicture 349

Removed:

- TRVStyle.SaveCSSFile

3 Overview

Parts of Documents

- TRichView item types 90
- TRichView items' "checkpoints" 92
- TRichView items' "tags" 96
- TRichView hypertext overview 97
- TRichView paragraphs 100
- Borders and background of paragraphs 102
- Styles and style templates 103
- Hidden text 106

Basic Operations

- Building TRichView document 107
- Scrolling in TRichView 110
- Selecting part of TRichView document 112
- TRichView Clipboard functions 113
- Searching and replacing in TRichView and TRichViewEdit 115
- Obtaining items of RichView 116
- Modifying TRichView items 117

Editing

- TRichViewEdit: Inserting items at position of caret 119
- Undo/redo in RichViewEdit 120
- Drag&drop 123

Graphics

- Animated images 124
- Smooth image scaling 125
- Semitransparent objects 126

Saving and Loading

- TRichView methods for saving and loading 127
- RichView Format (RVF) Overview 129
- Export to HTML 132

Internationalization

- Unicode [135]
- Bi-directional text [137]

Spelling Check

- Live spelling check [138]

Technical Information

- Viewer vs editor [140]
- Controls, documents, items [141]
- Valid documents [143]
- Units of measurement [144]
- Custom drawing [146]
- Cursors [147]
- RichView Format (RVF) specification [148]
- Lazarus [160]
- FireMonkey [161]
- Skia and Skia4Delphi [163]

3.1 Parts of documents

Overview [89] | Parts of Documents

- TRichView item types [90]
- TRichView items' "checkpoints" [92]
- TRichView items' "tags" [96]
- TRichView hypertext overview [97]
- TRichView paragraphs [100]
- Borders and background of paragraphs [102]
- Styles and style templates [103]
- Hidden text [106]

3.1.1 TRichView item types

Document in TRichView can contain items of the following types:

- **Text** [167] can have different fonts. Each text item is linked to one item in the collection of text attributes ("styles") [764]. 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 [135] encoding.
- **Tabulators** [168] are used instead of TAB (#9) characters.
- **Pictures** [169]: you can insert any Delphi graphic type in RichView. You can use TBitmap, TIcon, TMetafile, TJpegImage or third-party graphic classes [1136]. Gif images can be animated.
- **"Hot Pictures"** [172]: the same as pictures, but they are also hypertext links.

- **"Bullets"** [176] 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"** [178] are bullets-"hypertext links". They can specify two pictures: normal and "hot" (for highlighting effect under the mouse pointer)
- **"Breaks"** [173] are horizontal lines.
- **Controls** [174]: You can use any Delphi/C++Builder control merged in text.
- **List Markers** [180] represent paragraph bullets & numbering.
- **Tables** [179] arrange subitems in rows and columns.
- **Labels** [181] are non-text items looking like a text.
- **Numbered sequences** [183] allow to insert numbered fields (like "Fig. 1", "Fig. 2", ...) in text.
- **Endnotes** [184].
- **Footnotes** [185].
- **Sidenotes** [186] are notes displayed in floating boxes.
- **Text boxes** [188]
- **References to parent note** [190]
- **"Page number" fields** [190]
- **"Page count" fields** [192]
- **Mathematical expressions** [193] **[VCL]**

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") [757] 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 [129]). You can also export document as HTML, RTF, DocX file or print it (see export to html [132]).

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 [899].

See also: detailed overview of item types [164]

3.1.2 "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 [96])
- *RaiseEvent* flag; if *RaiseEvent*=True then TRichView generates OnCheckpointVisible [379] 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 [129];
- 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 [247]. 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 [743] for "normal" checkpoints (*clGreen* by default)
- RichView.Style.CheckpointEvColor [743] for checkpoints with *RaiseEvent=True* (*clLime* by default)

You can create your own procedure for drawing checkpoints, see TRVStyle.OnDrawCheckpoint [776].

Using checkpoints

A checkpoint is usually used to get its vertical coordinate (relative to the top of the document area) and scrolling the document so that the item associated with this 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 [271] 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 [360] method, and remove it by RemoveCheckpoint [340].

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[302], GetNextCheckpoint[315](CheckpointData), GetLastCheckpoint[313], GetPrevCheckpoint[319](CheckpointData) – allow you to obtain a list of all checkpoints in the document (except for checkpoints in table[196] cells, see below);
- GetItemCheckpoint[305](ItemNo) – returns the checkpoint associated with the specified item;
- GetCheckpointByNo[298](No) – returns the checkpoint with the specified index;
- FindCheckpointByName[294](Name) – returns the first checkpoint having the specified Name;
- FindCheckpointByTag[295](Tag) – returns the first checkpoint having the specified Tag;
- event OnCheckpointVisible[379] (Sender: TRichView; CheckpointData: TCheckpointData); this event is generated when:
  - RichView.CPEventKind[233] = *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[299](CheckpointData, Tag, Name, RaiseEvent) – returns Tag, Name and "RaiseEvent" flag of checkpoint;
- GetCheckpointXY[300](CheckpointData, X,Y), GetCheckpointYEx[301](CheckpointData): TRVCoord[1064] – obtaining information about coordinates of checkpoint (Y is the most useful; you can pass this Y to RichView.ScrollTo[875] method);
- GetCheckpointItemNo[299](CheckpointData) : Integer – returns the index of the associated item; it can return -1 for checkpoint in the end of document (no associated item);
- GetCheckpointNo[299](CheckpointData): Integer – returns the index of the checkpoint;

Methods belonging to the both groups:

- GetCheckpointY[300](no): TRVCoord[1064] – equivalent to GetCheckpointYEx(GetCheckpointByNo(No));

Methods of editor

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

- SetCheckpointInfoEd[560] – adds checkpoint for the specified item, as an editing operation;
- RemoveCheckpointEd[550] – 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[514] – returns checkpoint associated with the item at the caret position;

- SetCurrentCheckpointInfo[563] – adds/modifies checkpoint for the item at the caret position, as an editing operation;
- RemoveCurrentCheckpoint[551] – 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[525] – 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[512] – returns the checkpoint exactly at the caret position;
- RemoveCheckpointAtCaret[550] – deletes the checkpoint exactly at the caret position, as an editing operation.

RTF, DocX, and HTML

In HTML[132], checkpoints are saved[342] as anchors (<a name> tag). Either checkpoint name or checkpoint index can be used, see HTMLSaveProperties[244].UseCheckpointNames[445]. Links to checkpoints must start with '#' character. When importing HTML, anchors are imported as checkpoints. Optionally, "id" attributes of tags can be imported as checkponts, see HTMLReadProperties[243].IDAsCheckpoints[434].

In RTF and DocX, checkpoints are saved as bookmarks (checkpoints names are used). When importing RTF or DocX, a bookmark start is read as checkpoint (bookmark name is written in checkpoint name). When reading RTF or DocX, links to bookmarks are converted to links started from '#'. When writing RTF or DocX, 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[1096];
    Name: TRVUnicodeString[1099];
    RaiseEvent: Boolean;
    CheckpointData: TCheckpointData;
begin
  CheckpointData := MyRichView.GetFirstCheckPoint[302];
  while CheckpointData<>nil do begin
    MyRichView.GetCheckpointInfo[299](CheckpointData,Tag,Name,RaiseEvent);
      // processing this checkpoint
    ...
    CheckpointData := MyRichView.GetNextCheckpoint[315](CheckpointData);
  end;
end;
```

Example 2

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

```
procedure EnumCheckpoints(RVData: TCustomRVData[1014]);
var i,r,c: Integer;
  table: TRVTableItemInfo[901];
```

```
  Tag: TRVTag[1096];
  Name: TRVUnicodeString[1099];
  RaiseEvent: Boolean;
  CheckpointData: TCheckpointData;
begin
  for i := 0 to RVData.ItemsCount-1 do begin
    CheckpointData := RVData.GetItemCheckpoint[305](i);
    if CheckpointData<>nil then begin
      RVData.GetCheckpointInfo[299](CheckpointData,Tag,Name,RaiseEvent);
      // processing this checkpoint
      ...
    end;
    if RVData.GetItem[304](i) is TRVTableItemInfo[901] then
    begin
      table := TRVTableItemInfo[901](RVData.GetItem[304](i));
      for r := 0 to table.RowCount[920]-1 do
        for c := 0 to table.ColCount[914]-1 do
          if table.Cells[912][r,c]<>nil then
            EnumCheckpoints(table.Cells[912][r,c].GetRVData[964]);
    end;
  end;
end;
```

Call:

```
EnumCheckpoints(MyRichView.RVData[254]);
```

In the second example, RVData.GetCheckpointXY[300] 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[875]), 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[743], OnDrawCheckpoint[776]

See also properties of TRichView:

Options[247],.CPEventKind[233]

See also events of TRichView:

OnCheckpointVisible[379]

See also methods of TRichView:

AddCheckpoint[271], SetCheckpointInfo[360], RemoveCheckpoint[340],·GetFirstCheckpoint[302], GetNextCheckpoint[315], GetItemCheckpoint[305], GetCheckpointByNo[298], FindCheckpointByName[294], FindCheckpointByTag[295], GetCheckpointInfo[299], GetCheckpointXY[300], GetCheckpointYEx[301], GetCheckpointItemNo[299], GetCheckpointNo[299], GetCheckpointY[300]

See also methods of TRichViewEdit:

GetCurrentCheckpoint[514], SetCheckpointInfoEd[560], RemoveCheckpointEd[550]

3.1.3 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"[92] also have tags; all rules for items tags work for checkpoints tags;
- table cells also have tags[963]; 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[483];
- 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[107] which do not have Tag parameter. These functions create new items with Tag=''.

Unicode note: tags are Unicode strings.

Example:

```
var Tag: TRVTag[1096];
    S: TRVUnicodeString[1099];
// Converting the first text item and its tag string to upper case
  MyRichView.GetTextInfo[321](0, S, Tag);
  MyRichView.SetItemText[368](0, UpperCase(S));
  MyRichView.SetItemTag[367](0, UpperCase(Tag));
```

Tags in older version of TRichView

In TRichView versions prior to v13.2, tags were Integers. Depending on *rvoTagsArePChars* Option[247], 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[277], AddBreak[269], AddPicture[278], AddHotspot[275], AddBullet[270], AddControl[272], AddBreak[269]

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

AddCheckpoint[271].

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

GetItemTag[310], GetBreakInfo[296], GetBulletInfo[297], GetHotspotInfo[303], GetPictureInfo[318], GetControlInfo[301], GetTextInfo[321].

TRichView methods for modifying items (including tags):

SetItemTag[367], SetBreakInfo[358], SetBulletInfo[359], SetHotspotInfo[364], SetPictureInfo[370], SetControlInfo[361].

TRichView methods allowing work with checkpoints tags:

FindCheckpointByTag[295], GetCheckpointInfo[299], SetCheckpointInfo[360].

TRichViewEdit methods for obtaining information about item in caret position:

GetCurrentTag[522], GetCurrentBreakInfo[512], GetCurrentBulletInfo[513], GetCurrentHotspotInfo[515], GetCurrentPictureInfo[522], GetCurrentControlInfo[515], GetCurrentTextInfo[523].

TRichViewEdit methods for inserting:

InsertStringTag, InsertStringATag, InsertStringWTag[539]

TRichViewEdit methods for modifying items (including tags):

SetItemTagEd[574], SetBreakInfoEd[557], SetBulletInfoEd[558], SetHotspotInfoEd[570], SetPictureInfoEd[576], SetControlInfoEd[560].

TRichViewEdit methods for modifying information about item in caret position:

SetCurrentTag[570], SetCurrentBreakInfo[561], SetCurrentBulletInfo[562], SetCurrentHotspotInfo[565], SetCurrentPictureInfo[569], SetCurrentControlInfo[564]

TRichViewEdit methods for working with checkpoints tags:

SetCheckpointInfoEd[560], SetCurrentCheckpointInfo[563].

TRichViewEdit options affecting tags:

EditorOptions[483].

3.1.4 Hypertext

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[823];
- tabs[168] and label items[181], linked to hypertext styles[823];
- *hot pictures*[172]
- *hotspots*[178]

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

- changing color of text items (TFontInfo.HoverColor[812], HoverBackColor[811]);
- underlining text items (TFontInfo.HoverEffects[812]);

- changing mouse cursor (any hypertext text style can have its own cursor, see TRVStyle.JumpCursor[752], TFontInfo.JumpCursor[813])
- for hotspots[178] 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[403], OnJump[390]) and cursor indication do not work in editor (TRichViewEdit[469]). 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[483].

Since version 1.6, hypertext works in ReadOnly[488] editor just like in viewer (since version 13, this behavior can be turned off using *rvoNoReadOnlyJumps* option from EditorOptions[483])

Besides, you can provide some specific actions in response on user clicking any RichView item, using OnRVMouseDown[402] and OnRVMouseUp[403] events.

Hypertext

Any hypertext link (text, *hotspot* or *hot picture*) has its own **id**. This **id** is passed in hypertext events - OnRVMouseMove[403] and OnJump[390]. The first hypertext item has **id** equal to RichView.FirstJumpNo[243], 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[403] 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[390].

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[312] 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[1014];
    ItemNo: Integer;
begin
  MyRichView.GetJumpPointLocation[312](id, RVData, ItemNo);
  URL := RVData.GetItemText[310](ItemNo);
  ShellExecute(Application.Handle, 'open', PChar(URL),
    nil, nil, SW_NORMAL);
end;
```

In the example above, we use GetJumpPointLocation[312] 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[310] method to recognize the item type ("style"), and next use GetTextInfo[321] or GetHotspotInfo[303] /GetPictureInfo[318]. 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[96]).

Example

```
uses ..., CRVFData, ShellApi;
...
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var URL: String;
    RVData: TCustomRVFormattedData[1014];
    ItemNo: Integer;
begin
  MyRichView.GetJumpPointLocation[312](id, RVData, ItemNo);
  URL := RVData.GetItemTag[310](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(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, cell, or RVData of inplace editor.

See also...

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

3.1.5 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.ParaStyles).

For the list of properties available for paragraphs, see TParaInfo 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 is not applied to the first line after the break;
- no additional spacing between this line and the previous line (SpaceBefore and SpaceAfter) is applied;
- if the paragraph has border/background, 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* or tables).

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

These methods are:

```
procedure AddNL(s, StyleNo, ParaNo, Tag);
procedure AddFmt(FormatStr, Args, StyleNo, ParaNo);
```

```
procedure AddTextNL(s, StyleNo, FirstParaNo, OtherParaNo,
  DefAsSingleParagraph, Tag);
procedure AddPicture(Name, gr, ParaNo, VAlign, Tag);
procedure AddHotspot(Name, ImageIndex, HotImageIndex,
  ImageList, ParaNo, VAlign, Tag);
procedure AddBullet(Name, ImageIndex, ImageList, ParaNo, VAlign, Tag);
procedure AddControl(Name, ctrl, ParaNo, VAlign, Tag);
procedure Add(s, StyleNo); // equivalent to AddNL(s, StyleNo, -1).
```

See also: building a document, item types.

See also methods of TRichView:

```
function LoadText(FileName, StyleNo, ParaNo, AsSingleParagraph,
  CodePage):Boolean;
function LoadTextW(FileName, StyleNo, ParaNo,
  DefAsSingleParagraph):Boolean;
function AppendRVFFromStream(Stream, ParaNo):Boolean;
function IsParaStart(ItemNo): Boolean;
function IsFromNewLine(ItemNo): Boolean;
function GetItemPara(ItemNo): Integer;
```

Text flow around left- and right-aligned items

Some items (for example, pictures) can be aligned to the left or the right side (see TRVVAlign). 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 and/or ClearRight properties.

Editing

TRichViewEdit has a group of Insert*** methods inserting new item(s) at 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: Integer;
```

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

When the user moves the caret to paragraph with different style, OnCurParaStyleChanged 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(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(UserData: Integer);
```

See also...

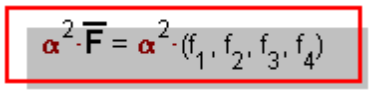
Borders and Background of Paragraphs [102]

3.1.6 Borders and background of paragraphs

Paragraphs can have borders and/or background color. These properties are defined in paragraph attributes ("styles") (see TParaInfo [828]). Border is defined in Border [831] property, background is in Background [830] property.

By default, there is no border around paragraph (TRVStyle.ParaStyles[i].Border.Style [854]=*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.

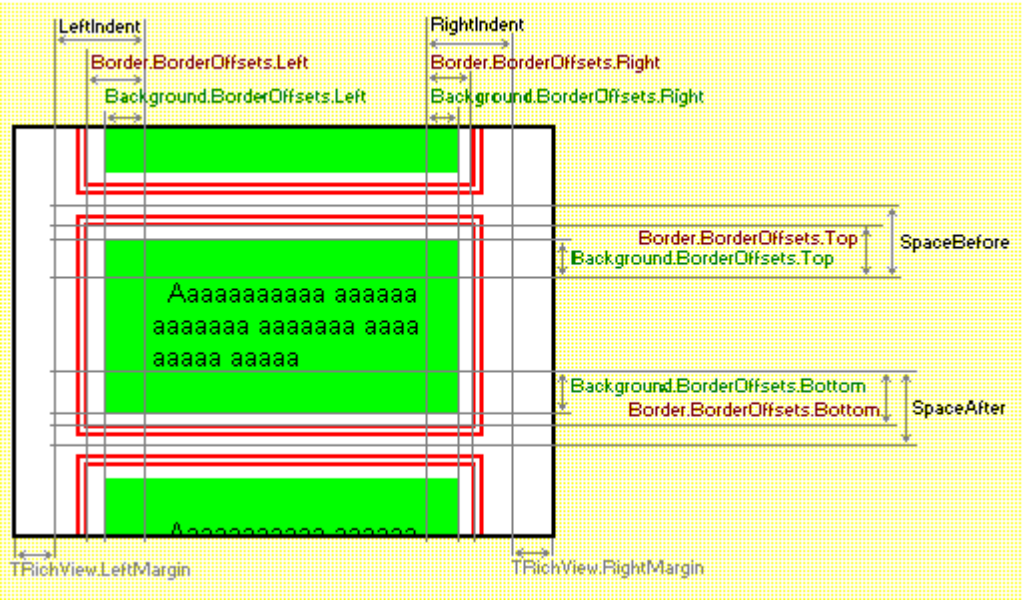
Top, right, bottom, left offsets (i.e. gap and padding) are defined by Top, Right, Bottom, Left properties of BorderOffsets.

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 [836] property of the same paragraph.

BorderOffsets.Bottom must not exceed SpaceAfter [835] property of the same paragraph.



Scheme: properties of paragraph border, background and layout

You can hide some edges of border using Border.VisibleBorders [854] property.

Breaks [173] 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 [831]

TRVBorder [852]

3.1.7 Styles and styleTemplates

As you could see, this manual sometimes calls items in TextStyles [764] and ParaStyles [757] "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 [764] and ParaStyles [757], and they are not changed while the application works.

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

Example

In this example, RichView1: TRichView [216] is linked to RVStyle1: TRVStyle [739].

```
RVStyle1.TextStyles.Clear;
with RVStyle1.TextStyles.Add do begin
```

```
  StyleName[805] := 'Normal Text';
end;
with RVStyle1.TextStyles.Add do begin
  StyleName := 'Title Text';
  Size[817] := 14;
  Style[817] := [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[829] := rvaCenter;
  Options[833] := [rvpaoKeepWithNext];
end;
...
RichView1.Clear[286];
RichView1.AddNL[277]('Chapter 1', 1, 1);
RichView1.AddNL('Once upon a time...', 1, 1);
RichView1.Format[295];
```

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[810] := clRed;
  FontName[811] := 'Verdana';
end;
with RVStyle1.ParaStyles[1] do begin
  Alignment := rvaLeft;
end;
RichView1.Reformat[339];
```

You can configure[224] 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[764] and ParaStyles[757]; 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[501], ApplyParaStyleConversion[499]). RichViewActions add new items as well.

In this mode, items of TextStyles[764] and ParaStyles[757] 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[764] and ParaStyles[757] may be marked as Standard[804] (usually, the items added initially). They may have a special meaning, they are not removed by DeleteUnusedStyles[293],

they can be saved in RTF as a style sheet (if *rvrtfSaveStyleSheet* in RTFOptions [252]). 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 [762] is a collection in TRVStyle [739] component. Unlike TextStyles [764] and ParaStyles [757], style templates do not define text and paragraph properties of document items [163] and paragraphs directly. Instead, they provide a mechanism for modifying TextStyles [764] and ParaStyles [757].

Using style templates is optional; they are used only if TRichView.UseStyleTemplates [263]=*True*.

Document items do not have direct links to style templates, they are still linked with TextStyles [764] and ParaStyles [757]. But items in TextStyles [764] and ParaStyles [757] may be linked to style templates using StyleTemplateId [805] 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 [167], tab [168], label [181], sequence [183], footnote [185], endnote [184], or note reference [190]).

It's recommended (but not necessary) to link every paragraph style to some style template. The recommended name [845] 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 [845] of style templates: applicable to text, applicable to paragraphs, applicable to text and paragraphs.

Each style template may have properties of both text [847] and paragraph [846]. Unlike items of TextStyles or ParaStyles, a style template:

- may define only a subset of text and paragraph properties (listed in ValidTextProperties [848] and ValidParaProperties [847]);
- may inherit text and paragraph properties from a parent style template [847].

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 [764][StyleNo].StyleTemplateId [805], a paragraph style template as TextStyles [764][StyleNo].ParaStyleTemplateId [815]. TRichView automatically provides for each document item that TextStyles [764][StyleNo].ParaStyleTemplateId [815] = ParaStyles [757][ParaNo].StyleTemplateId [805]. 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 [764] and ParaStyles [757] 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 [762] are edited.

TRichViewEdit has the following methods for applying the specified style template to the selection: ApplyStyleTemplate[501], ApplyTextStyleTemplate[502], ApplyParaStyleTemplate[500]. The same methods may be used to clear a style template or additional formatting.

Style templates can be edited using ChangeStyleTemplates[508] method.

3.1.8 Hidden text

Hidden items

Some items can be hidden. Hidden items are displayed only if *rvoShowHiddenText* is included in the Options[247] property of TRichView[216] control (they are displayed with a special dotted underline).

Text items[167] are hidden, if *rvteoHidden* is included in the Options[814] property of their style[821].

Non-text items[90] are hidden, if a non-zero value is assigned to *rvepHidden* extra item integer property[1067].

Import and Export

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

RTF (Rich Text Format) and DocX: hidden items are saved as hidden text and objects. Loading depends on the value of RTFReadProperties[253].SkipHiddenText[466] property of TRichView[216] control. By default (*False*), hidden text and objects are not loaded from RTF/DocX (skipped).

HTML: The CSS version of HTML export (HTMLSaveProperties[244].HTMLSavingType[441] = *rvhtmlstNormal)* saves hidden items as hidden text and objects. The simple version of HTML export saves hidden items like normal items. Loading depends on the value of HTMLReadProperties[243] .SkipHiddenText[435] property of TRichView[216] control. By default (*False*), hidden text and objects are not loaded from HTML (skipped).

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.

Markdown: when saving Markdown, hidden text is saved only is TRichView.MarkdownProperties.SaveHiddenText[456] = *True*.

Performance

A text formatting procedure is somewhat faster, if the option for displaying hidden items (*rvoShowHiddenText* in Options[247]) is set.

3.2 Basic Operations

Overview[89] | Basic Operations

- Building TRichView document[107]
- Scrolling in TRichView[110]
- Selecting part of TRichView document[112]
- TRichView Clipboard functions[113]
- Searching and replacing in TRichView and TRichViewEdit[115]

- Obtaining items of RichView [116].
- Modifying TRichView items [117].

3.2.1 Building TRichView 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 [286];
<Methods for adding contents>
MyRichView.Format [295];
```

Note: all methods introduced in RichView (except for the formatting methods) do not reformat document, do not redraw RichView and do not call OnChange [582] 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 [167]

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) [277] – adds one item with text *S*; *StyleNo* is an index in the collection of text styles (TRichView.Style.TextStyles [764]); *ParaNo* is an index in the collection of paragraph styles (RichView.Style.ParaStyles [757]) (if this item will start a new paragraph) or -1 (if this item will continue the paragraph). Optionally, you can specify Tag [96].
- Add(S, StyleNo) [277] – equivalent to AddNL(S, StyleNo, -1).
- AddTextNL(s, StyleNo, FirstParaNo, OtherParaNo, Tag) [281] – 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 [96].
- AddFmt(FormatStr, Args, StyleNo, ParaNo) [273] adds Format(FormatStr, Args) string, where Format is a function from SysUtils unit.

Methods working with ANSI/Unicode explicitly

- AddNLA, AddNLW[277], AddTextNLA, AddTextNLW[281]

Appending Images[169]

You can add picture of any Delphi format – TBitmap, TIcon, TMetafile, TJpegImage or third party graphic format[1136].

You should create a picture and TRichView will destroy it itself when necessary (for example, when you call RichView.Clear[286]).

AddPicture(Name, gr, ParaNo, VAlign, Tag)[278], where:

- Name: TRVUnicodeString[1099] – name of picture; this string is not used by TRichView itself; this string should not contain cr and lf characters;
- gr: TRVGraphic[1032] - 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[96].

Appending *Hot Images*[172]

- AddHotPicture[274]

Appending Delphi Controls[174]

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).

AddControl(Name, ctrl,ParaNo, VAlign, Tag)[272] adds a control, just like AddPicture adds a picture.

VCL and LCL: TRichView uses Tag properties of inserted components for internal needs (do not confuse with TRichView items tags).

Appending *Bullets*[176] – Images from ImageLists

AddBullet (Name, ImageIndex, ImageList, ParaNo, VAlign, Tag)[270], where:

- Name: TRVUnicodeString[1099] – 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[96].

Appending *Hotspots*[178] – 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)[275] 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*[173] – Horizontal Lines

- AddBreak.[269]

Adding Paragraphs' Bullets and Numbering[180] (list markers)

List markers can be added using SetListMarkerInfo[369] 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[179], labels[181], numbered sequences[183], endnotes[184], footnotes[185], references to parent footnotes and endnotes[190], page numbers[190], page counts[192], equations[193], custom item types[195] do not have special methods for adding them to documents. All of them are added by AddItem[276] method.

Appending *Checkpoints*

Checkpoints are not items of TRichView, but the method for adding *checkpoints* is similar to methods for adding items: AddCheckpoint[271]. See "checkpoints"[92].

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

Appending and Inserting RVF[129]

You can append RVF data by the methods:

- function InsertRVFFromStream[323](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[283](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 [757]) (if item will start a paragraph) or -1 (if the item will continue paragraph).

Appending Content of Another TRichView

- procedure AppendFrom [282](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 [127] (**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 [281]), but there is a method that affects their behavior: TRichView.SetAddParagraphMode [357].

Some custom item types (such as tables [196]) do not have special methods for appending them. They can be appended by general method AddItem [276].

3.2.2 Scrolling

Vertical scrolling [VCL and LCL]

Documents in TRichView 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 [264] 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 [295] method).

You can get or set position of vertical scrollbar using VScrollPos [874] property (in range 0..VScrollMax [874])

You can also use method ScrollTo [875] 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 [874] property.

DocumentHeight[239] 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[262] and BottomMargin[231].

You can get vertical coordinate in document for:

- *checkpoints*[92]
- hypertext link[97] (GetJumpPointY[313])
- any item (GetItemCoords[306])

Vertical scrolling [FireMonkey]

Vertical scrollbar appears automatically when necessary.

You can get or set position of vertical scrollbar using VScrollPos[874] property (in range 0..VScrollMax[874])

You can also use method ScrollToPosition[875] for scrolling to the specified vertical position, measured in pixels.

DocumentHeight[239] 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[262] and BottomMargin[231].

You can get vertical coordinate in document for:

- *checkpoints*[92]
- hypertext link[97] (GetJumpPointY[313])
- any item (GetItemCoords[306])

Horizontal scrolling

Horizontal scrollbar appears automatically when necessary.

In VCL and LCL versions, you can hide it with HScrollVisible[872] property

You can get or set the position of horizontal scrollbar using HScrollPos[872] property (in range 0..HScrollMax[871])

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[245], RightMargin[251])

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

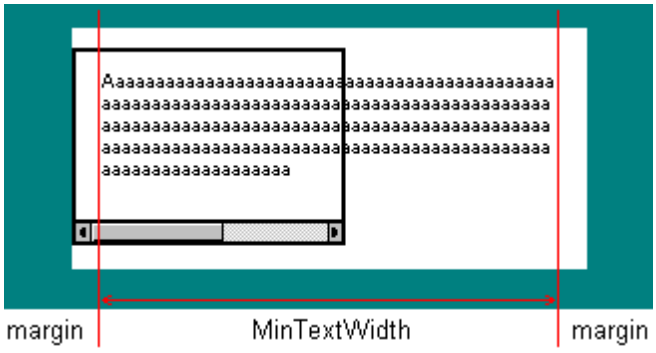
```
max(client width of TRichView,
    "maximal width of pictures or controls" +
    "maximal left[832] + first line[831] +  right[835] indent" + margins),
```

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

Text in paragraphs with no-wrap option[833] can also affect to width of scrolling area, in the same was as wide pictures or controls.

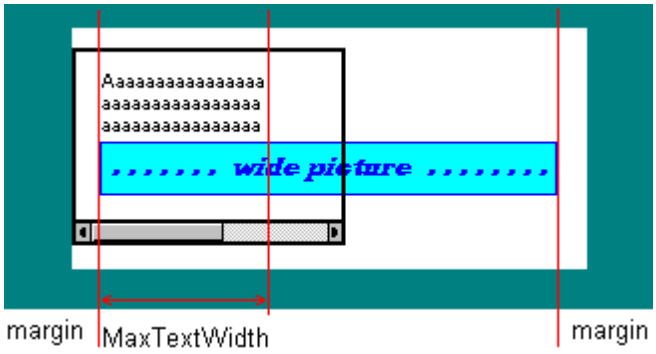
Use MinTextWidth[246] property to set the minimal width for text wrapping (text wraps when its length in pixels exceeds this value). Horizontal scrollable area cannot be less than

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[246] 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[247]. 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[833] for paragraph style. If this option is set, this paragraph will not be wrapped automatically.

3.2.3 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[758].

Selection Appearance

Selected text has RichView.Style.SelTextColor/InactiveSelTextColor[751] color and RichView.Style.SelColor/InactiveSelColor[751] background.

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

See also RVStyle.SelectionStyle[759].

Operations

You can:

- disallow making selection (clear *rvoAllowSelection* in Options[247]); please do not use in editor!
- determine if the selection is not empty (SelectionExists[356]);
- deselect TRichView (Deselect[294]; Invalidate);
- select the whole document (SelectAll[355]; Invalidate);
- get selection bounds (GetSelectionBounds[320]);
- select a part of the document (SetSelectionBounds[372];Invalidate);
- select the inserted control (SelectControl[355]);
- get the selection as a text (GetSelText, GetSelTextW[321]);
- get the selection as a Rich Text Format (SaveRTFToStream[351], SaveRTF[350]);
- get the selection as an image, if image is selected (GetSelectedImage[319]);
- save the selection to stream or file (SaveRVFToStream[352], SaveRVF[351]);
- copy the selection to the Clipboard in various formats (see RichView Clipboard Functions Overview[113])

In editor you can:

- delete selection (DeleteSelection[511]);
- select the current word[555] or line[554];
- cut selection to the Clipboard (CutDef[511]);

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[115]

3.2.4 Clipboard

TCustomRichView and its descendants can copy information to the Clipboard as a plain text, an image, RTF and RVF, Markdown, and your own custom formats.

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

Copying [VCL; Lazarus; FMX for Windows, macOS, and Linux]

The main method for copying to the Clipboard is CopyDef[288]. It can copy selection in one or more formats.

This method copies data in formats specified in TRichView.Options[247]:

Option	Meaning
rvoAutoCopyText	if set, CopyDef copies selection as ANSI text

Option	Meaning
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[288] copies selection to the Clipboard in all formats;
- CopyTextA[290] copies the selection as ANSI text (deprecated; Windows only);
- CopyTextW[290] copies selection as Unicode text
- CopyImage[289] – if there is a selected image, this method copies it to the Clipboard;
- CopyRVF[290] copies selection in RVF format[129]. 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[129], RVF Specification[148])
- CopyRTF[290] copies selection as RTF (Rich Text Format).
- CopyMarkdown[289] copies selection as Unicode text in Markdown format.

SelectionExists[356] method answers to the question "does the selection contain some data (not empty)?"

OnCopy[381] event allows copying in your own formats.

Copying [FMX for Android and iOS]

In FireMonkey for mobile platforms, only one format can be copied to the Clipboard. Copy[288] and CopyDef[288] work identically: they copy the most appropriate format from the formats listed in *rvoAutoCopy** Options[247].

The options *rvoAutoCopyText* and *rvoAutoCopyUnicodeText* work identically: Unicode text is copied.

Pasting

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

Pasting:

- PasteBitmap[544] pastes bitmap from the Clipboard, if available;
- PasteMetafile[546] pastes metafile picture from the Clipboard, if available [VCL];
- PasteGraphicFile[544] pastes graphic file(s) the Clipboard, if available [Windows];
- PasteText[548] pastes text from the Clipboard, if available;
- PasteRVF[547] pastes RVF from the Clipboard, if available;
- PasteRTF[547] pastes RTF from the Clipboard, if available;
- PasteHTML[545] pastes HTML from the Clipboard, if available;
- PasteURL[549] pastes URL from the Clipboard, if available [Windows];
- Paste[543] tries to paste as an RVF; if cannot – as an RTF, if cannot – as HTML, 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[478] property is taken into account. This method is called automatically when user press Ctrl+V or Shift+Insert.

- PasteMarkdown[546] pastes text from the Clipboard as Markdown, if available.

Testing is the Clipboard has the specific format:

- CanPaste[507] – "does the Clipboard contain any format what can be pasted in RichViewEdit?"
- CanPasteRVF[507] – "does the Clipboard contain RVF?"
- CanPasteRTF[507] – "does the Clipboard contain RTF?"
- CanPasteHTML[507] – "does the Clipboard contain HTML?"

OnPaste[594] 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[511] copies the selection (like CopyDef) and then deletes it (like DeleteSelection[511]). 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[123]

3.2.5 Searching and replacing

The following main functions for text searching are available:

```
function TRichView[216].SearchTextW[354](s: TRVUnicodeString[1099];
  SrchOptions: TRVSearchOptions[1091]): Boolean;
function TRichViewEdit[469].SearchTextW[553](s: TRVUnicodeString[1099];
  SrchOptions: TRVESearchOptions[1066]): Boolean;
```

These methods search for the substring **s**. If found:

- if OnTextFound[418] event is assigned, this event is called;
- otherwise, or if DoDefault parameter of OnTextFound[418] = *True*, the methods select this substring and make it visible (scroll the window to it).

A selection requires a formatted document, so, unless the result of text searching is processed inside OnTextFound[418], text searching methods require a formatted document as well.

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.

The last stored search result position can be used instead of selection or a caret position. This position is stored by StoreSearchResult[375] method, and is used as a starting point if *rvsroFromStored/rvseoFromStored* is included SrchOptions.

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(fo: TFindOptions): TRVSearchOptions;
function GetRVESearchOptions(fo: TFindOptions):TRVESearchOptions;
```

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

3.2.6 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,...). But there is one important exception: when using tables, **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 returns the number of items.

```
function RichView.GetItemStyle(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).

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
rvsPicture	-3	picture
rvsHotspot	-4	*hotspot*
rvsComponent	-5	Delphi control
rvsBullet	-6	*bullet*
rvsHotImage	-10	*hot-picture*
rvsListMarker	-11	list marker (bullet or numbering of paragraphs)

Constant	Value	Meaning
rvsTab	-12	tabulator [168]
rvsTable	-60	table [179]

See also: rvsXXX constants [1127])

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

- GetBreakInfo [296],GetBulletInfo [297], GetHotspotInfo [303], GetPictureInfo [318], GetControlInfo [301], GetTextInfo [321].
- For item of any type your can get additional property: GetItemExtraIntProperty [307] and GetItemExtraStrProperty [308].
- For item of any type your can get "tag" (GetItemTag [310]), "checkpoint" (GetItemCheckpoint [305])
- For any item you can get information about its paragraph: (GetItemPara [309], IsParaStart [325], IsFromNewLine [325]).
- For any item you can get its text (text for text item, name for non-text item) (GetItemText [310])
- For any item you can get object representing it in a document (GetItem [304], usually used for tables [901])

In editor you can use the methods above or GetCurrentBreakInfo [512], GetCurrentBulletInfo [513], GetCurrentHotspotInfo [515], GetCurrentPictureInfo [522], GetCurrentControlInfo [515], GetCurrentTextInfo [523] (**GetCurrent***Info** methods), and GetCurrentTag [522], GetCurrentCheckpoint [514], GetCurrentItemText [519].

The arguments of these methods are similar to the arguments of **Add***Tag** methods for appending items [107].

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** (or **Set***InfoEd** for editor) [117]. Exception: items with *rvepShared* [1067] property.

A table at the position of the caret (or a top-level table containing the caret) can be obtained with the method GetCurrentItemEx [517].

3.2.7 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 [480],...) But there is one important exception – when using tables [196], **SetCurrent***Info** methods modify item at 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 [510] and vice versa [510])

First, you need to know a type of item. You can use RichView.GetItemStyle[310](ItemNo). Or you can use CurItemStyle[481] property of RichViewEdit for the item at 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[107].

For the specified items you can use

- ·SetBreakInfo[358], SetBulletInfo[359], SetHotspotInfo[364] (call Invalidate to repaint RichView after them, reformatting is not necessary)
- ·SetPictureInfo[370], SetControlInfo[361] return Boolean value: "reformatting[295] 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[367] method to change tags of items,
- SetCheckpointInfo[360] to change checkpoints of items,
- SetItemExtraIntProperty[366] to change values of additional integer properties,
- SetItemExtraStrProperty[367] to change values of additional string properties,
- SetItemText[368] to change item text (visible text for text items, names for non-text items).

Please read the topic about "tags"[96] if you want to change tags of items.

The limitations of this version are:

- you can't change ImageList of *bullets*[176] and *hotspots*[178];
- you can't replace control with the different one;

It is not recommended to use these methods in editor, see Viewer vs Editor[140]

Set***InfoEd methods (editor)

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

SetBreakInfoEd[557], SetBulletInfoEd[558], SetHotspotInfoEd[570], SetPictureInfoEd[576], SetControlInfoEd[560];

also SetItemTagEd[574], SetCheckpointInfoEd[560], SetItemTextEd[574], SetItemExtraIntPropertyEd[572], SetItemExtraStrPropertyEd[573]

SetCurrent***Info methods (editor)

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

SetCurrentBreakInfo[561], SetCurrentBulletInfo[562], SetCurrentHotspotInfo[565], SetCurrentPictureInfo[569], SetCurrentControlInfo[564],

also SetCurrentTag[570], SetCurrentCheckpointInfo[563], SetCurrentItemText[568], SetCurrentItemExtraIntProperty[566], SetCurrentItemExtraStrProperty[567].

Additional methods for controls

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

```
procedure ResizeControl(ItemNo: Integer; NewWidth, NewHeight: TRVCoord);
procedure ResizeCurrentControl(NewWidth, NewHeight: TRVCoord);
```

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(ItemNo: Integer);
procedure AdjustControlPlacement2(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.

3.3 Editing

Overview | Editing

- TRichViewEdit: Inserting items at position of caret
- Undo/redo in RichViewEdit
- Drag&drop

3.3.1 Inserting at the position of caret

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

1. check ReadOnly 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 event, set Modified 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 property):

```
procedure InsertText(text, CaretBefore);
```

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

```
function InsertTextFromFile(FileName, CodePage):Boolean;
function InsertTextFromFileW(FileName):Boolean;
function InsertOEMTextFromFile(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(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*** ** methods [107]):

```
function InsertControl[526](Name, ctrl, VAlign): Boolean;
function InsertPicture[534](Name, gr, VAlign): Boolean;
function InsertHotPicture[528](Name, gr, VAlign): Boolean;
function InsertBreak[523](Width, Style, Color): Boolean;
function InsertBullet[524](ImageIndex, ImageList): Boolean;
function InsertHotspot[528](ImageIndex, HotImageIndex, ImageList): Boolean;
```

Inserting Files and Streams

These methods insert RVF [129] file or stream

```
function InsertRVFFromStreamEd[538](Stream):Boolean;
function InsertRVFFromFileEd[537](FileName):Boolean;
```

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

```
function InsertRTFFromStreamEd[536](Stream):Boolean;
function InsertRTFFromFileEd[536](FileName):Boolean;
```

These methods insert text of the current text style (see CurTextStyleNo [482] property):

```
function InsertTextFromFile[541](FileName):Boolean;
function InsertTextFromFileW[541](FileName):Boolean;
function InsertOEMTextFromFile[533](FileName):Boolean;
```

Inserting Tables and Other Items

The most general method for inserting items is

```
function InsertItem[531](Name, Item): Boolean;
```

It's recommended to use this method only for tables [179], labels [181], numbered sequences [183], endnotes [184], footnotes [185], references to parent footnotes and endnotes [190], page numbers [190], page counts [192], equations [193], and custom item types [195].

See also...

Methods for pasting from the Clipboard [113].

3.3.2 Undo and redo

(Introduced in version 1.3)

All methods for document editing that introduced in editor (TRichViewEdit [469]) can be undone/redone (see editor-style methods).

All methods for document modification that introduced in TRichView [216] 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[490]: 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[286] method, or when you use methods for loading documents (which call Clear inside).

You can clear undo and redo buffer yourself using method ClearUndo[509].

Implementing menu/toolbar buttons for undo/redo

You can undo/redo operations from the application code, using methods Undo[577] and Redo[549].

You can change enabled/disabled state of menu items or buttons comparing the returned values of functions UndoAction[577] and RedoAction[549] with *rvutNone* (*rvutNone* – menu for undo/redo should be disabled; other values – enabled).

Call these methods inside OnChange[582] 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[577] /RedoAction[549].

There are no predefined names for standard undo/redo actions. The Editor demo (Demos*\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[578]/RedoName[550] 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[505] to give a standard name, or BeginUndoCustomGroup[504] to give a custom name.

Next, you need to call SetUndoGroupMode[577].

```
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[1098], except for *rvutNone* and *rvutCustom*.
This value will be returned by UndoAction/RedoAction when undo/redo.

```
procedure TRichViewEdit.BeginUndoCustomGroup(const Name: TRVUnicodeString[1099]);
```

– 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[489])

Example (inserting picture and resizing it 100x100):

```
MyRichViewEdit.TopLevelEditor.BeginUndoGroup(rvutInsert);
MyRichViewEdit.TopLevelEditor.SetUndoGroupMode(True);
try
  if MyRichViewEdit.InsertPicture[534](...) then
  begin
    MyRichViewEdit.SetCurrentItemExtraIntProperty[566](
      rvepImageHeight, 100, True);
    MyRichViewEdit.SetCurrentItemExtraIntProperty(
      rvepImageWidth, 100, True);
  end;
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.

Grouping operations for undo, an alternative way

There is one more way to group operation for undo. It is less efficient, but allows grouping operations in multiple table cells.

Use BeginUndoGroup2..EndUndoGroup2[505] to give a standard name to the group, or use BeginUndoCustomGroup2..EndUndoCustomGroup2[504] to give a custom name. Unlike the methods described in the previous section, these methods should be called for the root editor, not for cell inplace editors.

Example (replace all occurrences of FindText ro ReplaceText):

```
MyRichViewEdit.SetSelectionBounds[372](
  0, MyRichViewEdit.GetOffsBeforeItem[317](0),
  0, MyRichViewEdit.GetOffsBeforeItem(0));
MyRichViewEdit.BeginUndoGroup2(rvutInsert);
try
  while MyRichViewEdit.SearchText[553](FindText, [rvseoDown, rvseoMulti
    MyRichViewEdit.InsertText[540](ReplaceText);
finally
  MyRichViewEdit.EndUndoGroup2(rvutInsert);
```

```
end;
```

See also...

Methods introduced in TRichView and undo: see Viewer vs Editor [140]

3.3.3 Drag and drop

You can drag data from TRichView [216] or TRichViewEdit [469]. You can drop data to TRichViewEdit [469].

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 [VCL, LCL]

TRichView provides data in the following formats:

- RichView Format [129] ('RichView Format')
- RTF ('Rich Text Format')
- ANSI text (CF_TEXT)
- Unicode [135] text (CF_UNICODETEXT)
- bitmap (CF_BITMAP), if TBitmap picture [169] is selected
- metafile (CF_ENHMETAFILE), if TMetafile picture [169] is selected.

You can disallow dragging from the given TRichView by including *rvoDisallowDrag* in the Options [247] property.

TRichView cannot initialize dragging of inserted controls [174]. If you want to drag them, use BeginOleDrag [284] method.

Dragging to TRichViewEdit [VCL, LCL]

AcceptDragDropFormat [478] lists formats that can be accepted.

By default all formats are accepted except for URL (because it requires processing in OnReadHyperlink [395])

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 [483], 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 [247], file name is is stored in *rvespImageFileName* extra string property [1072] for pictures; you can modify this file name in OnAssignImageFileName [379] event.

The following events may occur:

- OnReadHyperlink [395];
- OnDropFile [586];
- OnDropFiles [586].

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

- OnOleDragEnter [589];

- OnOleDragOver[590];
- OnOleDragLeave[589];
- OnOleDrop[591],
- OnBeforeOleDrop and OnAferOleDrop[580]

Drag&Drop in FireMonkey

When dragging between TRichView/TRichViewEdit controls within the same application, data are transferred using the most appropriate format from AcceptDragDropFormat[478].

When dragging from TRichView to other target, data are provided as text or as an image (if an image is selected).

When dragging from TRichViewEdit, data are accepted as files, text, or an image.

See also...

Clipboard[113]

3.4 Graphics

Overview[89] | Graphics

- Animated images[124]
- Smooth image scaling[125]
- Semitransparent objects[126]

3.4.1 Animated images

Overview

TRichView can animate inserted images[169].

AnimationMode[228] property defines when animation starts. By default, AnimationMode[228] =*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[291]. If you simply load and edit documents, set this mode to *rvaniOnFormat*.

Methods:

- StartAnimation[374] starts all animations;
- StopAnimation[375] stops all animations;
- ResetAnimation[341] 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

Gif animation for older versions of Delphi.

Downloads: ***https://www.trichview.com/resources/thirdparty/gifimage.zip***.

Include RVGifAnimate unit in your project to enable this animation.

3. RVJvGifAnimate for TJvGifImage from JEDI's JVCL

Downloads: ***https://github.com/project-jedi/jvcl***.

4. Grid animation

An image 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[1067] type. If at least one of *rvepImageWidth* and *rvepImageHeight* is not defined, animation is not played.

This feature works if RVGridAnimate unit is included in the project (add it to "uses" of the main form unit).

3.4.2 Smooth image scaling

Scaling [VCL and LCL]

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

The following images are affected:

- TCustomRichView.BackgroundPicture[229] (if the image is scaled in BackgroundProperties)[229];
- pictures[169] and *hot-pictures*[172] (when stretched);
- table.BackgroundImage[904] (if the image is scaled in BackgroundProperties[905]);
- table cells' BackgroundImage[953] (if the image is scaled in BackgroundProperties[954]).

The settings for scaling can be changed using RVThumbnailMaker[1129] object.

Smooth scaling is applied when displaying pictures on the screen. It is not used when printing.

Sometimes, smooth scaling is undesirable (for example, for scaling pixel art or bar codes). You can turn off/on smooth scaling for the specific graphic item using *rvepSmoothScaling*[1067] property.

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.

FireMonkey note: in FireMonkey, TRichView uses the standard TBitmap scaling method that usually produces good results.

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[1114] scaled images (+ a scaled version TCustomRichView.BackgroundPicture[229], which is always stored).

Smooth metafile drawing [VCL]

VCL version of TRichView can draw inserted TMetafile images using GDI+ functions. GDI+ provides higher quality of drawing, including anti-aliased lines.

To use this feature, add RVGDIPlusGrIn unit in your project (or include it in "uses" of the main form's unit).

3.4.3 Semitransparent objects

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

- pictures [169] (for example, Gif, Png, icons, Bitmaps with alpha-channel)
- paragraph background (with specified Opacity [852])
- table [179] background (that may include transparent pictures [904] and semitransparent [916] color filling)
- table cell background (that may include transparent pictures [953] and semitransparent [958] color filling)
- sidenote [186] and text box [188] background (with specified Opacity), displayed only while printing in TRVPrint [696] and in ScaleRichView.

FireMonkey note: in FireMonkey, any colors can be semitransparent, so any object can be semitransparent. Opacity properties are applied in additional to color transparency.

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 [124] and backgrounds of inplace editors of table cells.

Known problems: TPngObject does not support semitransparency when displayed rotated [961].

Printing in TRVPrint [696] [VCL and LCL]

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).

FireMonkey note: in FireMonkey, TRichView does not use special methods for printing semitransparent objects, because the standard drawing produces good results.

Drawing and printing in TRVReportHelper [721]

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[723] 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.

3.5 Saving and loading

Overview[89] | Saving and Loading

- TRichView methods for saving and loading[127]
- RichView Format (RVF) Overview[129]
- Export to HTML[132]

3.5.1 Saving and loading

TRichView has the following methods for export:

- SaveHTML[342], SaveHTMLToStream[346] – exporting the document to a file or a stream as HTML;
- SaveText[352]: saving plain ANSI text in file; SaveTextW[352]: saving plain Unicode text; SaveTextToStream, SaveTextToStreamW[353] – the same for streams (can save the whole document or only a selection);
- SaveMarkdown[348], SaveMarkdownToStream[349] – saving the whole document or the selected part to ***Markdown*** file or stream;
- SaveRTF[350], SaveRTFToStream[351] – exporting the whole document or the selected part to a file or a stream as RTF (Rich Text Format);
- SaveDocX[341], SaveDocXToStream[342] – exporting the whole document or the selected part to a file or a stream as DocX (Microsoft Word format);
- SaveRVF[351], SaveRVFToStream[352] – saving the whole document or the selected part to RVF (RichView Format[129]) file or stream.

TRichView has the following methods for import:

- LoadHTML[330], LoadHTMLFromStream[332] – reading HTML file or stream;
- LoadText[338]: reading plain ANSI text; LoadTextW[338]: reading plain Unicode (UTF-16) text;
- LoadTextFromStream, LoadTexFromStreamtW[339] – the same for streams instead of files;

- LoadMarkdown[332], LoadMarkdownFromStream[333]: reading ***Markdown*** file or stream;
- LoadRTF[334], LoadRTFFromStream[335] – reading RTF (Rich Text Format) file or stream.
- LoadDocX[326], LoadDocXFromStream[327] – reading DocX (Microsoft Word Document) file or stream.
- LoadRVF[336], LoadRVFFromStream[337] – reading RVF (RichView Format[129]) file or stream.

Universal methods:

- LoadFromFile[328], LoadFromStream[329] – reading from a file or a stream, a format is detected by content.

See also: InsertRVFFromStream[323], AppendRVFFromStream[283].

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

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

- InsertRVFFromFileEd[537], InsertRVFFromStreamEd[538] – like InsertRVFFromFile/InsertRVFFromStream, but editor-style methods.
- InsertRTFFromFileEd[536], InsertRTFFromStreamEd[536] – inserting RTF (Rich Text Format) file or stream.
- InsertDocXFromFileEd[526], InsertDocXFromStreamEd[527] – inserting DocX (Microsoft Word Document) file or stream.
- InsertTextFromFile[541] – editor-style method;
  InsertOEMTextFromFile[533] – the same method for files in DOS code page;
  InsertTextFromFileW[541] – the same method for Unicode files;
- InsertMarkdownFromFileEd[532], InsertMarkdownFromStreamEd[532] – inserting ***Markdown*** file or stream.

There are also method for pasting data from the Clipboard[113].

Events on export

- HTML: OnHTMLSaveImage[383], OnSaveImage2[410], OnSaveComponentToFile[405], OnSaveHTMLExtra[409], OnSaveItemToFile[411], OnSaveParaToHTML[413], OnWriteHyperlink[418], OnWriteObjectProperties[420].
- RTF: OnSaveComponentToFile[405], OnSaveRTFExtra[414], OnSaveItemToFile[411], OnWriteHyperlink[418], OnWriteObjectProperties[420].
- DocX: OnSaveComponentToFile[405], OnSaveDocXExtra[407], OnSaveItemToFile[411], OnWriteHyperlink[418], OnWriteObjectProperties[420].
- Text: OnSaveComponentToFile[405], OnSaveItemToFile[411]
- Markdown: OnSaveImage2[410], OnSaveComponentToFile[405], OnWriteHyperlink[418].

Events on import

- Markdown: OnImportPicture[386], OnReadHyperlink[395], OnAddStyle[377].
- RTF, DocXL: the events above + OnStyleTemplatesChange[417].
- HTML: the events above + OnImportFile[385].

Common events

- OnProgress[392]

See also...

Saving HTML[132]
Reading/writing RVF[129]

Export of tables [209]
Unicode in TRichView [135]

3.5.2 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 [254]).

This format is used:

- in TDBRichView [604], TDBRichViewEdit [616] to save their contents to binary fields;
- RVF can be copied to the Clipboard; editor can paste RVF;
- data can be dragged&dropped [123] as RVF;
- you can load/save RVF with the methods: LoadRVFFromStream [337], InsertRVFFromStream [323], AppendRVFFromStream [283], LoadRVF [336], SaveRVFToStream [352], SaveRVF [351].

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 [224];
- in order to load *bullets* [176], *hotspots* [178] and list levels [859] with imagelist from RVF you need to process OnRVFImageListNeeded [400] event;
- you can save pictures [169] and controls [174] 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 [764] and paragraph [757] (i.e. styles) is saved in RVF documents optionally, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions [254];
- style templates are saved in RVF, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions [254], and UseStyleTemplates [263]=*True*.

RVF and Image Lists

TRichView can save and load all document content itself, but there is an exception: *bullets* [176], *hotspots* [178] and list markers having imagelist picture in list levels [859].

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 [400] 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* [176] or *hotspot* [178]. 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[96]. 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[401] and OnRVFPictureNeeded[401] events.

Example

```
procedure TMyForm.MyRichViewRVFControlNeeded(
  Sender: TCustomRichView; Name: TRVUnicodeString[1099];
  const Tag: TRVTag[1096]; 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[1096] – TRichView item tag (do not confuse with tag properties of TComponent descendants);
- Name: TRVUnicodeString[1099] – string that can be associated with any control or picture in RichView.

Example [VCL and LCL]

```
// this example assumes that images may be only bitmaps
procedure MyRichViewRVFPictureNeeded(Sender: TCustomRichView[216];
    const ItemName: TRVUnicodeString[1099]; Item: TRVNonTextItemInfo[899];
    Index1, Index2: Integer; var gr: TRVGraphic[1032]);
begin
  gr := TBitmap.Create;
  if (Index1>=0) and (Index2>=0) then
    gr.LoadFromFile((Item as TRVTableItemInfo[901]).Cells[912][Index1, Index
  else if Item is TRVTableItemInfo then
    gr.LoadFromFile(TRVTableItemInfo(Item).BackgroundImageFileName[905])
  else if Item is TRVGraphicItemInfo then
    gr.LoadFromFile(TRVGraphicItemInfo[899](Item).ImageFileName[1072]);
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[254], 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[254] define options for RVF saving and loading.

After loading you can check RichView.RVFWarnings[258] 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, twips, or EMU[144]) than specified in Style[739].Units[764] of TRichView which loads this file.

When inserting RVF, lengths from RVF are converted to Style[739].Units[764].

When loading RVF, the control works according to *rvfoCanChangeUnits* option. If it is excluded from RVFOptions[254], lengths from RVF are converted to Style[739].Units[764]. If it is included, and RVF

contains either text or paragraph styles, existing styles are converted to units read from RVF (using Style[739].ConvertToDifferentUnits[767] method), so lengths from RVF are read without conversion.

If a conversion of units is performed while reading RVF, *rvfwConvUnits* is included in RVFWarnings[258].

See also...

RVF Specification[148]

RichView properties:

- Options[247],
- RVFOptions[254],
- RVFWarnings[258];
- UseStyleTemplates[263];
- StyleTemplateInsertMode[260].

RichView events:

- OnRVFImageListNeeded[400],
- OnVFControlNeeded[400],
- OnRVFPictureNeeded[401],
- OnControlAction[380] (ControlAction=*rvcaAfterRVFLoad*),
- OnItemAction[387] (ItemAction=*rviaInserting* and *rviaInserted*);
- OnStyleTemplatesChange[417].

RichView methods:

- LoadRVFFromStream[337], InsertRVFFromStream[323], AppendRVFFromStream[283], LoadRVF[336],
- SaveRVFToStream[352], SaveRVF[351],
- Copy[288], CopyDef[288], CopyRVF[290].

RichViewEdit methods:

- InsertRVFFromStreamEd[538], InsertRVFFromFileEd[537],
- Paste[543], PasteRVF[547], CanPasteRVF[507], CutDef[511].

3.5.3 Export to HTML

Overview

TRichView has methods for saving, loading, and inserting HTML files.

The following properties are used for HTML saving:

- HTMLSaveProperties[244]
- DocParameters[237].Title[427] and Comments[424].

Images

TRichView saves HTML file and a set of image files (alternatively, images may be embedded directly in HTML file).

Web image formats (like Gif or Png) are saved as they are. Other images are saved as Jpegs (*.jpg).

You can specify graphic formats that must not be converted to Jpegs using RVGraphicHandler[1125].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[383] or OnSaveImage2[410] event (see demo in Demos*\Assorted\Save HTML\).

See also HTMLSaveProperties[244].ImageOptions[442], ImagesPrefix[443].

Two ways of HTML saving

There are two ways to save HTML in TRichView, defined HTMLSaveProperties[244].HTMLSavingType[441]:

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

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.

Checkpoints

Checkpoints[92] are saved as anchors.

By default, names are checkpoints are used as names of anchors, but you can switch to generated anchor names.

See HTMLSaveProperties[244].UseCheckpointNames[445], CheckpointsPrefix[437].

Encoding

By default, TRichView creates UTF-8 HTML files.

UTF-8 is highly recommended, but you can change encoding using HTMLSaveProperties[244].Encoding[439] property.

Saving Using CSS

By default, if HTMLSaveProperties[244].HTMLSavingType[441] = *rvhtmlstNormal*, CSS is saved as a style sheet in the <head> part of HTML, and referred in tags using "class" attribute.

Alternatively, you can store a style sheet in an external file, and specify its name in HTMLSaveProperties[244].ExternalCSSFileName[440].

Alternatively, instead of using a style sheet, styles can be saved directly in HTML tags (using "style" attribute).

Even if HTML is generally saved without CSS (HTMLSaveProperties[244].HTMLSavingType[441] = *rvhtmlstSimplified)*, you can store non-text items (such as tables, images or horizontal lines) with CSS.

See HTMLSaveProperties[244].CSSOptions[438].

You can provide additional styles in HTMLSaveProperties[244].ExtraStyles[440].

Hyperlinks, controls, more...

By default, items tags [96] are saved as targets of hypertext links. You can customize saving using the event OnWriteHyperlink [418].

By default, inserted controls are not saved. However, you can save them in the event OnSaveComponentToFile [405].

You can save additional information in HTML in the events: OnSaveHTMLExtra [409], OnSaveParaToHTML [413].

You can insert HTML codes [814] in text.

You can completely change saving of items of any type in HTML using OnSaveItemToFile [411] event.

You can save additional properties of non-text items using OnWriteObjectProperties [420] event.

Equation objects

If the free version of Adit Math Engine is used, equation objects [193] are exported as images.

If the free commercial of Adit Math Engine is used, equation objects [193] are exported as MathML objects (can be turned off by RVMathEquationManager [1054].SaveMathMLInHTML [1021] property).

Symbol Fonts

SaveHTML [342] saves all text of "Symbol" and "Wingdings" (and some other) fonts as a set of HTML character entities or Unicode codes. For example, the text 'a+b=g' of "Symbol" font ($\alpha+\beta=\gamma$) 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 [363] and/or a footer [362] 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 [237].TitlePage [427].

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 [127];

- Export of tables [209];
- Typed constants related to HTML export [1107];
- Example how to save MIME-encoded files (*.eml or *.mht), how to send HTML e-mail using TNMSMTP or TIdSMTP components: ***https://www.trichview.com/forums/viewtopic.php?t=11***.

3.6 Internationalization

Overview [89] | Internationalization

- Unicode [135]
- Bi-directional text [137]

3.6.1 Unicode

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 [338], LoadTextFromStream [339] load ANSI text files. A code page for conversion to Unicode is specified in the optional parameter.

LoadTextW [338], LoadTextFromStreamW [339] load Unicode text files.

Note: you can test file with the function

```
function RV_TestFileUnicode(const FileName: TRVUnicodeString[1099]): TRVUnicodeTest
```

defined in RVUni.pas.

Return values

- *rvutNo* – the file is not Unicode (odd size);
- *rvutYes* – the file is most likely Unicode (UTF-16) (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 [352] saves ANSI text file. Unicode strings are converted basing on Style.DefCodePage [745] property.

SaveTextW[352] 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***[132] 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[321] returns selection as an ANSI string. Unicode text is converted basing on Style.DefCodePage[745] property.

GetSelTextW[321] returns selection as a Unicode string.

Text searching methods have versions allowing to search for ANSI and for Unicode string: TRichView.SearchTextA/SearchTextW[354]; however, SearchTextA simply converts the string to Unicode (using Style.DefCodePage[745]) and calls SearchTextW.

CopyTextA[290] copies selection as ANSI text. Unicode strings are converted basing on Style.DefCodePage[745] property.

CopyTextW[290] copies selection as Unicode.

Copy[288] and CopyDef[288] are copy Unicode (Option[247]-rvoAutoCopyUnicodeText)

Editing Operations

If pasting text using Paste[543] method, and text is available in Clipboard, the method pastes Unicode text.

PasteTextA[548] pastes ANSI text, PasteTextW[548] pastes Unicode text.

InsertTextFromFile[541]: the file must be ANSI (converted, if needed)

InsertOEMTextFromFile[533]: the file must be OEM (converted, if needed)

InsertTextFromFileW[541]: the file must be Unicode (converted, if needed)

InsertText[540], InsertStringTag[539] add Unicode string in Delphi/C++Builder 2009+ and ANSI string in older versions of Delphi/C++Builder.

InsertTextA[540], InsertStringATag[539] add ANSI string (converted, if needed)

InsertTextW[540], InsertStringWTag[539] add Unicode string (converted, if needed)

RVF (RichView Format[129])

Applications compiled with older versions of TRichView (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 [258]: *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 [468].

3.6.2 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 [230] property of TRichView either to *rvbdLeftToRight* or to *rvbdRightToLeft*.

BiDiMode of document can be overridden by BiDiMode of paragraph [831], BiDiMode of paragraph can be overridden by BiDiMode of text item [808].

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 [VCL and LCL]

The results depends on the value of TextEngine [763]:

- if Windows API is selected, in *rvbdUnspecified* mode, 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.

Demo project

Demos*\Assorted\International\RTL\

3.7 Spelling check

Overview [89] | Spelling Check

- Live spelling check [138]

3.7.1 Live spelling check

A live spelling check marks misspelled words with special underlines:

- [VCL and LCL] wavy line of TRVStyle.LiveSpellingColor[755] color
- [FMX] dashed line of TRVStyle.LiveSpellingColor[755] color

A spelling check is executed in background thread.

Starting and finishing spelling check

In TRichView[216]:

Spelling check is started by calling StartLiveSpelling[374] method.

In TRichViewEdit[469]:

Spelling check is started:

- by calling StartLiveSpelling[374] method or
- when user modifies document or when any editing-style method is called (if LiveSpellingMode[487] =*rvlspOnChange*)

Spelling check is stopped by ClearLiveSpellingResults[286] method. Clear[286] also stops checking.

Using spelling checking components

The recommended component for spelling checking is TRVSpellChecker[728].

Additionally, the following third-party components are supported:

1. Addict 3 and 4 by Addictive Software (shareware Delphi component, discontinued)
2. ***ExpressSpellChecker*** by Developer Express Inc. (shareware Delphi component)
3. ***HunSpell*** (opensource C++ library under GPL/LGPL/MPL tri-license)
4. ***ASpell*** by Kevin Atkinson (opensource DLL, license: LGPL)
5. ***Polar SpellChecker Component by*** by Polar (shareware ActiveX)

The live spelling thread calls OnSpellingCheck[416] event for each word in the document.

"Word" is defined as a run of characters between delimiters[235]. Additionally, '&' is treated as a delimiter. Apostrophes are processed specially, see RichViewApostropheInWord[1111] global variable.

URLs are ignored, see RVIsCustomURL[1126] function.

[FMX] Using platform service

In FireMonkey, TRichViewEdit can use a spelling check service (IFMXSpellCheckerService) provided by some platforms (for example, macOS).

To enable using a platform service, assign CheckSpelling[480] = *True*.

In this case, it's not necessary to process OnSpellingCheck[416] event, words are checked using a platform service. However, if the event is assigned, it is used instead of a service.

Making corrections

Call LiveSpellingValidateWord[326] if user added word in a custom dictionary or ignore list.

Call GetCurrentMisspelling [521] 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 [521] to get misspelled word in the caret position and to generate suggestions;
2. when the user has chosen a suggestion, use GetCurrentMisspelling [521] again to select the misspelled word and use InsertText [540] to insert replacement.

Alternatively, misspelled word can be selected on the step 1 (before displaying popup menu).

Popup menu

VCL and LCL version of TRichView does not provide special methods for adding items for correction of misspelled word to a popup menu. However, they are implemented in ***RichViewActions***, in TRVAPopupMenu component, and in the components for TRichView+DevExpress integration.

In FireMonkey, if PopupMenu property is not assigned, TRichViewEdit displays a special default popup menu. Items for correction of misspelled words are added in this default menu automatically. If you use a third-party spelling checker instead of a platform service, OnGetSpellingSuggestions [382] event is used to add these items. Note that only items with suggestions are added ("Ignore", "Ignore All", "Add to Dictionary" items are not created automatically).

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 [286] and then create a new document [107], everything is OK, because Clear stops live spelling check. But if you make changes in existing document, call ClearLiveSpellingResults [286] 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)

3.8 Technical information

Overview [89] | Technical Information

- Viewer vs editor [140]
- Controls, documents, items [141]
- Valid documents [143]
- Units of measurement [144]
- Custom drawing [146]
- Cursors [147]
- RichView Format (RVF) specification [148]
- Lazarus [160]
- FireMonkey [161]
- Skia and Skia4Delphi [163]

3.8.1 Viewer vs editor

Methods of TRichView

TRichView[216] 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[277], AddPicture[278], LoadRTF[334].

Examples of the second group are SetPictureInfo[370], SetItemText[368].

Examples of the third group are: LoadRVF[336], LoadRTFFromStream[335].

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[295] (or FormatTail[296]) to display documents created or modified with such methods.

Table cells[951] 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[286] or ClearLiveSpellingResults[286]).

Methods of TRichViewEdit

TRichViewEdit[469] has the following groups of methods for modifying documents:

1. all methods of TRichView;
2. methods for inserting items at the position of caret (**Insert***** methods, such as InsertText[540], InsertPicture[534]);
3. methods for modifying the item at the position of caret (**SetCurrent***** methods, such as SetCurrentPictureInfo[569], SetCurrentItemText[568]);
4. extended set of methods for modifying item with the given index (**Set***Ed** methods, such as SetPictureInfoEd[576], SetItemTextEd[574]).

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[488] they do nothing; they also respect protection of text[815] and paragraphs[833];
- they call OnChange[582] event;

- they can be safely called while a live spelling is running;
- modifications can be undone and redone [120].

In an editor, the main application of the methods of the group 1 is generating new documents from scratch (after calling Clear [286]).

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 [286]; (this method also clears undo and redo buffers);
2. One or several calls of methods of the group 1;
3. Format [295] (after the document has been formatted, you can apply methods of the groups 2-4)

See also: Undo in tables [208]

3.8.2 Controls - documents - items

There are three main families of classes.

1. VCL controls, descendants of TCustomRichView.

See the diagram of class hierarchy [37].

The main visual controls are: TRichView [216], TRichViewEdit [469], TDBRichView [604], TDBRichViewEdit [616].

2. Documents, objects containing items, descendants of TCustomRVFormattedData [1014].

Documents are objects, but not components.

Each RichView or RichViewEdit has such document accessible as RVData [254] 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 [901] have cells [951]. Cells are also documents containing items, cells are descendants of TCustomRVFormattedData.

3. Items of documents, descendants of TCustomRVItemInfo [899].

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 [901]).

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 [964] (if cell is not edited, it is the cell itself; if edited, it is RVData [254] 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[1014]);
var i,r,c: Integer;
    s: String;
    table: TRVTableItemInfo[901];
begin
for i := 0 to RVData.ItemCount[244]-1 do
  if RVData.GetItemStyle[310](i)>=0 then
  begin
    // this is a text item
    s := RVData.GetItemText[310](i);
    s := AnsiUpperCase(s);
    RVData.SetItemText[368](i,s);
  end
  else if RVData.GetItemStyle[310](i) is TRVTableItemInfo[901] then
  begin
    table := TRVTableItemInfo[901](RVData.GetItem[304](i));
    for r := 0 to table.RowCount[920]-1 do
      for c := 0 to table.ColCount[914]-1 do
        if table.Cells[912][r,c]<>nil then
          AllUpperCase(table.Cells[912][r,c].GetRVData[964]);
  end;
end;
```

Call:

```
AllUpperCase(MyRichView.RVData);
MyRichView.Format[295];
```

Example 2

The same task, using different methods

```
// This procedure will be called for each RichView item
procedure TForm1.EnumItemsProc(RVData: TCustomRVData[1014];
  ItemNo: Integer;  var UserData1: Integer;
  const UserData2: TRVUnicodeString[1099]; var ContinueEnum: Boolean);
var s: String;
begin
  if RVData.GetItemStyle[310](ItemNo)>=0 then begin
    s := RVData.GetItemText[310](i);
    s := AnsiUpperCase(s);
    RVData.SetItemText[368](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[254].EnumItems(EnumItemsProc, v, '');
MyRichView.Format[295];
```

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.

3.8.3 Valid documents

There are some limitations on document structure.

Internally, each document item[163] 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[469] always create a valid document (providing that it was valid before the operation).

But some operations in viewer[216] can create invalid documents (especially DeleteItems[291])

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[291] and SetAddParagraphMode[357])
2. The first item (in document or table cell) must start a paragraph.
3. Full-line items (*breaks*[173] and tables[179]) 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[180] 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[810] >0, they can be in any place of text.

RichViewActions includes RVNormalize unit containing NormalizeRichView procedure. This procedure can fix most of document problems.

3.8.4 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[817].

Twip (tw, "twentieth of a point") is defined as 1/20 of point. It equals to 1/1440 of inch.

English metric unit (EMU) is useful for representation of decimal fractions of inches and millimeters as integer values. 1 inch = 914400 EMU, 1 mm = 36000 EMU.

Pixel (px) – a single point in a raster image or device. To convert inch (or other absolute unit of measurement) to pixels, a pixel density (referred as "pixels per inch" (PPI) or "dots per inch" (DPI)) 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, DPI of values measured in pixels is defined in UnitsPixelsPerInch[765] property of TRVStyle[739] component. When rendering on a device (screen or paper), sizes are recalculated according to the device DPI as:

value * (device DPI) / UnitsPixelsPerInch[765]

Screen DPI can be read from Screen.PixelsPerInch, or it can be overridden in RichViewPixelsPerInch[1115] global variable.

Properties measured in points

Font size (Size[817] property of text styles[764])

Properties measured in "standard" pixels

1 "standard" pixel = 1/TRVStyle.UnitsPixelsPerInch[765] of an inch.

The following properties of TCustomRichView[216] are measured in "standard" pixels:

- LeftMargin[245], RightMargin[251], TopMargin[262], BottomMargin[231] – margins;
- MaxTextWidth[246] – maximal width for word wrapping;
- MinTextWidth[246] – minimal width for word wrapping (minimal width of scrollable area, not including margins).

Properties measured in TRVUnits[1099]

Properties of the type TRVLength[1081] are measured in TRVUnits[1099]. They are values of a floating point type (Extended) that can represent lengths measured in inches, centimeters, millimeters, picas, points, or "standard" pixels.

In TRichView, this type is used for:

- properties of TRichView[216].DocParameters[237]; they are measured in TRichView[216].DocParameters[237].Units[428].
- properties of TRVPrint[696]; they are measured in TRVPrint[696].Units[707].

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[1094]

Properties of the type TRVStyleLength[1094] are measured in TRVStyleUnits[1094]. They are values of an integer type that can represent lengths measured in "standard" pixels, twips, or EMU.

The property TRVStyle[739].Units[764] defines units for properties (and sub-properties) of this TRVStyle component as well as for properties of items[163] in documents loaded in linked[260] TRichView controls.

Properties of TRVStyle[739] measured in TRVStyleUnits[1094]

The following properties are measured in TRVStyleUnits[1094]:

- DefTabWidth[745];
- CharSpacing[809] of text style[821];
- FirstIndent[831], LeftIndent[832], RightIndent[835], SpaceBefore[836], SpaceAfter[835] of paragraph style[837];
- Position[857] of item in a tab stop collection[836] of paragraph style;
- Left, Right, Top, Bottom of background padding rectangle (Background[830].BorderOffsets[851]) of paragraph style;
- Left, Right, Top, Bottom of border padding rectangle (Border[831].BorderOffsets[853]) of paragraph style, as well as border's Width[854] and InternalWidth[853];
- MarkerIndent[866], FirstIndent[860], LeftIndent[864] in list levels[842] of list style[841].

Item[163] properties measured in TRVStyleUnits[1094]

The following items properties[1067] are measured in TRVStyleUnits[1094]:

- *rvepSpacing, rvepImageWidth, rvepImageHeight, rvepVShift* (the latter only if *rvepVShiftAbs* is nonzero)

The following table[179] properties are measured in TRVStyleUnits[1094]:

- BorderWidth[909], CellBorderWidth[910];
- BorderHSpacing[907], BorderVSpacing[908], CellHSpacing[911], CellVSpacing[913];
- CellHPadding[911], CellVPadding[913], CellPadding[912];
- HRuleWidth[916], VRuleWidth[922];
- BestWidth[906] (only if value is positive).

The following table cell[951] properties are measured in TRVStyleUnits[1094]:

- BestHeight[955];
- BestWidth[955] (only if value is positive).

A width of break[173] (horizontal line) is measured in TRVStyleUnits[1094] as well.

The following label item[181] properties are measured in TRVStyleUnits[1094]:

- MinWidth[971].

The following sidenote[186] properties are measured in TRVStyleUnits[1094]:

- BoxProperties[992].Width[996] and Height[995] (if WidthType and HeightType are “absolute”);
- BoxPosition[993].HorizontalOffset[999] and VerticalOffset[1003] (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[739].Units[764].

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[739].Units[764]=*rvstuPixels*), or as fractional values in points (if TRVStyle[739].Units[764]=*rvstuTwips*).

In RVF[129] (RichView Format) files, values are saved as TRVStyle[739].Units[764]. When loading, they are either converted to TRVStyle[739].Units[764], or vice versa, TRVStyle[739].Units[764] is changed. See “Units of measurement” in RVF Overview[129]. Alternatively, TRichView can be converted to units from RVF file, if *rvfoCanChangeUnits* is included in RVFOptions[254].

3.8.5 Custom drawing

There are the following events for custom drawing.

TCustomRichView[216]

- OnPaint[392] – drawing over RichView document.

TRVPrint[696]

- OnPagePrepaint[719] – drawing on page under the document;
- OnPagePostpaint[718] – drawing on page over the document;
- OnPrintComponent[887] – printing components.

TRVStyle[739]

- OnDrawCheckpoint[776] – drawing checkpoint;
- OnDrawPageBreak[778] – drawing page break;
- OnDrawParaBack[780] – drawing background of paragraph;
- OnDrawTextBack[783] – drawing background of text;
- OnApplyStyle[774] – applying text style to Canvas;
- OnApplyStyleColor[775] – applying the text style color to Canvas;
- OnDrawStyleText[781] – drawing text of the given style;
- OnStyleHoverSensitive[784] – returns "should component be repainted when user moves the mouse to a hypertext of the given style?"

TRVTableItemInfo[901]

- OnDrawBorder[949] – drawing table and cell borders.

3.8.6 Cursors

Definition and use

Constant	Used by	Meaning	Unit*
crJump	TRVStyle 739 component: ▪ TRVStyle.JumpCursor 752; ▪ TFontInfo.JumpCursor 813.	Hypertext	**RVStyle.pas**
crRVZoomIn	TRVPrintPreview 635 component (ZoomInCursor 895)	Zoom in	**CRVPP.pas**
crRVZoomOut	TRVPrintPreview 635 component (ZoomOutCursor 896)	Zoom out	**CRVPP.pas**
crRVSelectRow	TRVTableItemInfo 901	Table row selection	**RVTable.pas**
crRVSelectCol	TRVTableItemInfo 901	Table column selection	**RVTable.pas**
crRVFlipArrow	TRVStyle 739 component (LineSelectCursor 753)	Line selection	**RVStyle.pas**
crRVLayingIBeam	TRichViewEdit 469	Vertical text	**RVStyle.pas**

* in FireMonkey version, unit names start from "fmx", i.e. fmxRVStyle.pas instead of RVStyle.pas.

VCL and LCL

In Delphi VCL and in Lazarus, TRichView defines and uses the following custom mouse cursors:

Image	Constant	Value
*	*crJump*	101
	crRVZoomIn	102
	crRVZoomOut	103
	crRVSelectRow	104
	crRVSelectCol	105

Image	Constant	Value
	crRVFlipArrow	106
	crRVLayingIBeam	107

* in Windows2000+, TRichView uses the system default hypertext cursor instead (since version 1.4)

FireMonkey

In Delphi FireMonkey, TRichView uses only standard mouse cursors.

Constant	Value
crJump	*crHandPoint*
crRVZoomIn	*crArrow*
crRVZoomOut	*crArrow*
crRVSelectRow	*crUpArrow*
crRVSelectCol	*crUpArrow*
crRVFlipArrow	*crArrow*
crRVLayingIBeam	*crIBeam*

3.8.7 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 (129).

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*. Most 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[167]; <Item Type> is an index in the collection of styles (RVStyle.TextStyles[764])
< 0	*rvs****[1127] constants	non-text item[163] or special object (see the rows below)
-2	*rvsCheckpoint*	invisible label ("checkpoint"[92])
-7	*rvsBack*	background (image, color)
-8	*rvsVersionInfo*	RVF version (only in <Version Info>)
-9	*rvsDocProperty*	text, paragraph, list styles,layout information, DocProperties[238], DocParameters[237]

<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[757]) (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. TRichView 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* [96].

<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 [254])
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 [254]).
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 [401], OnRVFControlNeeded [400].

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 [254] (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[1099];
  Tag: TRVTag[1096]; 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[379] 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[167] (Items)

<Record Type> >= 0

<Rest of Header> := [<Additional Properties>] [<Hint>]

<Hint> := <quoted string>, value of rvespHint[1072] property.

<Additional Properties> := **+**<quoted string>

<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

<Additional Properties> are not stored by default. They can be used to store additional properties in subclassed TRVTextItemInfo[167].

Break[173] (Item)

<Record Type> = *rvsBreak*

<Paragraph> is ignored

<Rest of Header> ::= <Break Color><Break Style><Break Width> [<Color Opacity>]

<Break Color> ::= <int>

<Break Style> ::= <int>

<Break Width> ::= <int>

<Break Color> is a color of horizontal line, TColor; if Break Color=*clNone* then line will be of color of the 0th text style. FMX version may also save <Color Opacity>, a value from 0 to 255.

<Break Style> is TRVBreakStyle[1059] value converted to integer.

<Break Width> is a width of line.

<String Count> >= 0.

Picture[169] (Item)

<Record Type> = *rvsPicture*

<String Count> >= 3

<Rest of Header> = <Vertical Align>

<Vertical Align> ::= <int>

<Vertical Align> is a vertical alignment of picture (TRVVAlign[1100] 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'`) or a graphic file extension ('bmp', 'ico', 'wmf', 'jpg')

<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: [VCL and LCL] 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. If the class is not registered, TRichView allows to detect the image by format and choose the appropriate graphic class itself.

Note 2: Behavior of TRichView when loading unregistered graphics format is defined in RVFOptions[254] 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[258] property will contain *rvfwUnknownPicFmt*.

Control[174] (Item)

<Record Type> = *rvsComponent*

See the previous section, adjusted for: *rvfoIgnoreUnknownCtrls*, *rvfwUnknownCtrls*.

Bullet[176] (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[400] 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[254] property. If *rvfoConvLargeImageIdxToZero* option is defined then index is set to 0. Otherwise the function returns False (reading failed). In both cases RVFWarnings[258] property will contain *rvfwConvLargeImageIdx*.

Hotspot[178] (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[238]

<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[260].TextStyles[764])
2	Collection of paragraph styles (Style[260].ParaStyles[757])
3	Layout (margins, BiDiMode[230], MaxTextWidth[246], MinTextWidth[246]) of TRichView
4	Collection of list styles (Style[260].ListStyles[755])
5	DocProperties[238]
6	Special information, written by TRVReportHelper[721] when saving page

7	DocParameters [237]
8	Subdocument (headers and footers for TSRichViewEdit)
9	RVStyle.Units [764]
10	Collection of style templates (Style [260].StyleTemplates [762])
11	List of names of style templates
12	DocObjects [235]
13	GetReadDocumentPixelsPerInch [319] value

<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 [845] (<special string>), the second line contains its id [844].

For 12, DocObjects are saved using a special TRVCollectionBuilder collection, to allow storing items of different classes.

For 13, the next line contains an integer value (as text).

Background (old, obsolete version)

This is an obsolete record type. It can be read, but is not saved by new versions of TRichView.

<Record Type> = *rvsBack*

This item is similar to *rvsPicture*, but has differences:

- <Rest of Header> is a background color and style, and optionally color opacity
- 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> [<Color Opacity>]

<Background Style>::= <int> (TRVBackgroundStyle [229] value converted to integer)

<Color>::=<int> (background color)

FMX version may also save <Color Opacity>, a value from 0 to 255.

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>

Background color is saved as TColor, even in FireMonkey version. FireMonkey version may also save color opacity (as an integer value from 0 to 255).

Background (new version)

<Record Type> = *rvsBackEx*

<Rest of Header>::=<Color> [<Color Opacity>]

Background color is saved as TColor, even in FireMonkey version. FireMonkey version may also save color opacity (as an integer value from 0 to 255).

If no background image, and all background properties are default, <String Count> = 0. Otherwise, it is equal to the count of strings after the header:

- <Picture Delphi Type> | 'none'
- <Background Property>*
- <Picture Data>

<Picture Delphi Type> is a name of the background picture class (such as 'TBitmap', 'TIcon', 'TMetafile', 'TJPEGImage') or a graphic file extension ('bmp', 'ico', 'wmf', 'jpg').

<Picture Data> is the same as for picture items (see above). It is saved only if the first line after the header is not equal to 'none'.

<Background Property> is a property of BackgroundProperties.

<Background Property> ::= <PropertyName> '=' <PropertyValue>

<PropertyName> ::= 'HorizontalAlignment' | 'VerticalAlignment' | 'HorizontalSizeKind' | 'VerticalSizeKind' | 'HorizontalOffsetKind' | 'VerticalOffsetKind' | 'HorizontalOffset' | 'VerticalOffset' | 'ImageWidth' | 'ImageHeight' | 'Attachment' | 'HorizontalRepeat' | 'VerticalRepeat' | 'Visible'

<PropertyValue> ::= <int> (property value typecasted to integer)

List Markers [180] (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 [181] (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[972] property. <Text Style> ::= <int>, it is TextStyleNo[973] property. <Min Width> ::= <int>, it is MinWidth[971] property. <Alignment> ::= <int>, it is ord(Alignment[971]). The next line depends on ProtectTextStyleNo[972]. <Item Name> ::= <string>.

Sequence[183] (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[981] property. <Numbering Type> ::= <int>, it is NumberType[980] property. <Reset> ::= <int>, it is ord(Reset[981]). <Start From> ::= <int>, it is Reset[981] property. <Text Style>. The next five lines have the same meaning as for labels. <Formatting String> ::= <special string>, it is FormatString[980] property.

Footnote[185] and EndNote[184] (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[984] 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 [190] (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 [1067] and string [1072] 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>

Note that all color properties are stored as TColor values, even in FireMonkey (where all color properties have TAlphaColor type). In addition to the color itself, FireMonkey version can store an additional color opacity property (with value from 0 to 255).

This specifications does not cover a case of saving style names instead of style indexes (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 [238], extra string properties [1072], tag strings (if *rvoTagsArePChars* in Options [247]; note: since TRichView v13.2 tags are always strings), style names (if *rvfoUseStyleNames* in RVFOptions [254]) 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.

Changes in RichView version 23.3

rvsBack record is obsolete. *rvsBackEx* is used instead.

3.8.8 Lazarus

TRichView components can be used in Lazarus.

The following platforms are supported:

- Windows (32-bit and 64-bit)
- Linux (GTK2 widgetset + Cairo canvas for printing)

Currently, the following sets of components are ported to Lazarus:

- TRichView
- RichViewActions
- RvHtmlImporter (obsolete)
- RichViewXML
- ASpell parser (Windows only)
- HunSpell parser (Windows only)
- VirtualTree support
- ScaleRichView and SRVControls (Windows only)
- ReportWorkshop (Windows only)

TRVOfficeConverter[687] and TRVASpell use 32-bit DLL, so they can be only in Win32 applications.

Limitations

Common limitations

The following functions are not available in Lazarus:

- gestures (text selection and print preview zooming with fingers)
- mouse panning (scrolling by clicking a middle mouse button and then moving the mouse)
- gif and metafile images (unless you find Lazarus-compatible graphic components that is able to display, load and save these formats)
- when dragging an image from TRichView, it is not provided as a bitmap (however, TRichViewEdit still accepts bitmaps dragged into it)
- LiveBindings

Additional Linux limitations

The following functions are not available in Lazarus for Linux:

- advanced character properties (horizontal scaling, character spacing)
- distribute alignment for paragraphs (justification by increasing character spacing)
- cell rotation in tables
- drag and drop

A quality of print preview is low in Lazarus for Linux. It can be improved by assigning TRVPrintPreview[635].CachePageImage[638] = *True*.

Strings

UTF-16 strings

Most methods and properties in the components have TRVUnicodeString[1099] parameters, which is defined as UnicodeString in Lazarus. These string contain Unicode (UTF-16) text.

ANSI strings

Parameters in ***A methods have TRVAnsiString[1056] parameters, which is defined as AnsiString in Lazarus. They are treated as ANSI string (not UTF-8!) in Windows.

These methods include:

- in TRichView: AddNLA[277], AddTextNLA[281], GetItemTextA[310], GetSelTextA[321], GetWordAtA[322], SearchTextA[354], SetItemTextA[368]
- in TRichViewEdit: GetCurrentItemTextA[519], InsertStringATag[539], InsertTextA[540], SearchTextA[553], SetCurrentItemTextA[568], SetItemTextEdA[574]

UTF-8 strings

Some methods have String parameters. They are treated as UTF-8 in Lazarus

These methods include:

- in TRichView: AddFmt[273], AddNL[277], AddTextNL[281], GetItemText[310], GetSelText[321], GetWordAt[322], SearchText[354], SetItemText[368]
- in TRichViewEdit: GetCurrentItemText[519], InsertStringTag[539], InsertText[540], SearchText[553], SetCurrentItemText[568], SetItemTextEd[574]

Demo projects

TRichView demo projects for Lazarus are in:

- <TRichView Dir>\TRichView\Demos\Lazarus\

Some add-ins and additional components have Lazarus demo projects as well:

- <TRichView Dir>\TRichView\Demos\Addins\BlendBitmap\Demo\Lazarus\
- <TRichView Dir>\TRichView\Demos\Addins\ComboItem\Demo\Lazarus\
- <TRichView Dir>\RvHtmlImporter\Demos\Lazarus\
- <TRichView Dir>\ThirdParty\ASpell\Demos\Lazarus\
- <TRichView Dir>\ThirdParty\HunSpell\Demos\Lazarus\

3.8.9 FireMonkey

TRichView supports the following platforms in FireMonkey:

- Windows 32-bit and 64-bit (RAD Studio XE6 and newer)
- macOS 64-bit (Delphi 10.3 and newer)
- macOS ARM 64-bit (Delphi 11 and newer)
- Android 32-bit and 64-bit (RAD Studio 10.4 and newer)
- Linux 64-bit (Delphi 10.3 and newer + FMXLinux 1.74 or newer)
- iOS 64-bit (Delphi 10.4 and newer)
- iOS 64-bit ARM Simulator (Delphi 11 and newer)

Currently, the following sets of components are ported to FireMonkey:

- TRichView
- RvHtmlImporter (deprecated, use HTML loading methods instead)
- RichViewXML

Differences between VCL/LCL[160] version and FireMonkey version:

Feature	VCL and LCL	FireMonkey	Comments
Color type	TColor Semi-transparency is supported only for objects that have Opacity property (paragraph, table, cell backgrounds)	TAlphaColor Semitransparent colors are allowed anywhere.	You can use TRVColor[1063] type
Graphic type	Classes inherited from TGraphic: TBitmap, TPngImage, etc.	TRVGraphicFM[1032] Not supported: Windows Metafiles and GIF animations.	You can use TRVGraphic[1032] class
Font size type	Integer	Single	You can use TRVFontSize[1076] type
Coordinate type	Integer	Single	You can use TRVCoord, TRVCoordRect, TRVCoordPoint types[1064]
Windows system colors	Can be used (for example, *clWindow*)	Cannot be used. When reading files, these colors are mapped to the most appropriate TAlphaColor constant *(clNone* is mapped to *TAlphaColorRec.Null)*	
Text attributes	All	Not supported: character scaling, character spacing	
Paragraph attributes	All	Not supported: "distribute" alignment	
Mouse cursors	Custom cursors are used	Only standard mouse cursors are used	See about cursors[147].
Zooming in TRichView[216]	DocumentPixelsPerInch[239]	ZoomPercent[265]	

3.8.10 Skia and Skia4Delphi

Skia is an open source 2D graphics library developed by Google. It provides common APIs that work across a variety of hardware and software platforms. It serves as the graphics engine for Google Chrome and ChromeOS, Android, Flutter, and many other products.

Skia4Delphi is a cross-platform 2D graphics API for Delphi based on Google's Skia Graphics Library.

Skia4Delphi is included in RAD Studio starting from RAD Studio 12 Athens. For older versions of Delphi, ***it can be downloaded separately***.

When using Skia4Delphi, TRichView receives additional features.

VCL

In VCL version, Skia4Delphi adds support of additional graphic formats: SVG, WebP, WBMP. Graphics of these types can be inserted in TRichView.

To use SVG support at full extent, add **RVSkia** unit in your project.

FireMonkey

Skia canvases

In FireMonkey, Skia4Delphi provides its own canvases that can replace default FireMonkey canvases.

To support these canvases in TRichView, add **fmxRVSkiaFM** unit in your project.

Additionally, fmxRVSkiaFM unit allows using SVG images and PDF saving.

Known problem: drawing on Linux Skia canvas is slow, avoid using GlobalUseSkia = True with TRichView for Linux.

Graphics

Skia4Delphi adds several TBitmap codecs (although, some of them are decoding-only codecs).

Additionally, Skia4Delphi supports SVG images. They can be inserted in TRichView as TRVSvgImageSkiaFM[1032] class.

You can use RVGraphicHandler[1125].GetFileDialogFilter[1033] to get a filter for opening and saving dialogs. It returns a filter string containing not only raster formats (registered for TBitmap), but also additional formats (such as SVG), if they are available.

PDF

You can use TRVPrint.SavePDF[715] and TRVReportHelper.SavePDF[727] to create PDF files.

4 Overview of Items

Item types

- RichView Item Types[164]
- Text Items[167]
- Tabulators[168]
- Pictures[169]

- *Hot-Pictures* [172]
- *Breaks* [173]
- Controls [174]
- *Bullets* [176] (images from image-lists)
- *Hotspots* [178]
- Tables [179]
- List markers [180] (paragraph bullets & numbering)
- Labels [181] (non-text item looking like text)
- Numbered sequences [183]
- Endnotes [184]
- Footnotes [185]
- Sidenotes [186] (notes displayed in floating boxes)
- Text Boxes [188] (floating boxes)
- References to parent notes [190]
- "Page number" fields [190]
- "Page count" fields [192]
- Custom Item Types [195]

4.1 Overview

Items

TRichView documents consist of items. Items are indexed from 0 to ItemCount [244]-1. Some items may contain sub-documents which in their order contain items. Such items are:

- tables [196] (each cell is a sub-document displayed in the same TRichView control)
- notes (endnotes [184], footnotes [185], sidenotes [186]) and text boxes [188] (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 [244]-1.

Item Types

A type (also called "style" in this manual) identifier of the given item is returned by the method GetItemStyle [310] (in editor, a style of the item at the position of caret is returned by CurItemStyle [481]).

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 [764] collection of the the linked [260] TRVStyle [739] component.

If the returned value is negative, this is a constant identifying the item class.

The standard TRichView item types are:

- text [167]
- tabulators [168]
- pictures [169]
- *hot-pictures* [172] (pictures with hyperlink)
- *breaks* [173] (horizontal lines)
- controls [174] (any Delphi/C++Builder/Lazarus control)

- *bullets* [176] (images from image-list)
- *hotspots* [178] (bullets with hyperlink)
- tables [179]
- list markers [180] (paragraph bullets & numbering)
- labels [181] (non-text item looking like text)
- numbered sequences [183]
- endnotes [184]
- footnotes [185]
- sidenotes [186]
- references to parent notes [190]
- text boxes [188]
- mathematical expressions [193]

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 [135] encoding. ANSI text must not contain characters (CR (#13), LF (#10), #0, TAB (#9), FF (12).

See GetItemText [310], SetItemText [368] 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 [96]

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 [92]

Pagebreak-before Flag

If this flag is set, this item starts a new page when printing.

See PageBreaksBeforeItems [250].

Paragraph Attributes

Items are organized in paragraphs. Paragraph attributes are defined in the ParaStyles [757] collection of the linked [260] TRVStyle [739] component.

GetItemPara[309] returns an index of paragraph style for the given item. This method returns the same value for all items in the same paragraph.

IsParaStart[325] returns True, if the given item starts a new paragraph.

IsFromNewLine[325] 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[1100]).

You can move item up or down by the specified number of pixels/twips/EMU or percent. See the following extra item properties[1067]: *rvepVShift*, *rvepVShiftAbs*.

For text items, see VShift[820] property of text style.

Spacing

You can define spacing around the item. See the following extra item properties[1067]: *rvepSpacing*

Coordinates

GetItemCoords[306] returns coordinates of the top left corner of the given item relative to the beginning of the document.

GetItemClientCoords[306] 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[815] properties of text style.

You can protect non-text items from deletion. See the following extra item integer properties[1067]: *rvepDeleteProtect*.

Visibility

Items can be hidden[106]. Hidden items are displayed only if *rvoShowHiddenText* is included in the Options[247] property of TRichView[216] control.

Text items are hidden, if *rvteoHidden* is included in the Options[814] property of their style[821].

Non-text items are hidden, if a non-zero value is assigned to *rvepHidden* extra item integer property[1067].

Popup Hints

The component can show a hint message when the mouse pointer is above the item.

See the following extra item string properties[1072]: *rvespHint*.

Hint message can be dynamically customized in OnItemHint[388] event.

Events

OnItemAction[387] occurs when item is inserted, deleted, moved to undo list, or when item text is modified.

OnSaveItemToFile[411] 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.

4.2 Text items

Text Item

Text item displays text in traditional (ANSI) or Unicode[135] encoding.

Style[310] of this item type: >=0 (index in the TextStyles[764] collection of the linked[260] TRVStyle[739]).

Item of this collection (TFontInfo[821] object) completely defines all properties of this text.

Main properties of TFontInfo[821]:

- Jump[823] - determines if this text is hypertext[97] or not;
- properties defining font, colors, background, hypertext cursors, protection, bi-di mode[137], and others.

Text item cannot contain:

- CR and LF (#13 and #10) characters;
- TAB (#09) characters (they are inserted as tabulators[168] or replaced with several space characters, depending on SpacesInTab[761]);
- #0 characters;
- FF (#12) characters (see PageBreaksBeforeItems[250] property of TCustomRichView).

Methods of TCustomRichView[216]

The following viewer-style methods add item of this type to the end of the document:

- single item: AddNL[277] (and shortcut Add[277]), AddFmt[273]
- multiple items: AddTextNL[281].

The following viewer-style methods change text of the given item

- SetItemText -A -W[368]

The following methods return text of the given item:

- GetItemText -A -W[310]

Appending text from file/stream:

- LoadText, LoadTextW[338],
- LoadTextFromStream, LoadTextFromStreamW[339]

Methods of TCustomRichViewEdit[469]

The following editor-style methods insert item of this type at the position of caret:

- single item: InsertStringTag, InsertStringATag, InsertStringWTag[539];
- multiple items: InsertText, -A, -W[540].

The following editor-style methods change text of the given item:

- SetItemTextEd -A -W[574], SetCurrentItemText -A -W[568]

The following methods return text of item at the position of caret:

- GetCurrentItemText -A -W[519]

The following editor-style methods insert text from files at the position of caret:

- InsertTextFromFile, InsertTextFromFileW[541]

See Also...

See also methods for copying and pasting[113], selecting text[112].

Some of the methods above can insert tabulators[168] as well.

See also: classes for item types[899].

4.3 Tabulators

Tabulators

A special item used to align the subsequent text to the next tab stop.

Tab stops are measured from the LeftMargin[245] and defined by the following properties:

- paragraph's LeftIndent[832] (for paragraph with hanging (negative) FirstIndent[831]);
- paragraph's collection of Tabs[836]
- TRVStyle.DefTabWidth[745].

Style[310] of this item type: *rvsTab* (-12)

Properties of this item type are defined by:

- its text style (the same properties as for text items[167])
- corresponding tab stop (item in the paragraph's collection of Tabs[836]: TRVTabInfo[856])

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[761] property of the linked TRVStyle[739] 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[483] property.

Tabulators are read from RVF (RichView Format) files and streams regardless of value of SpacesInTab[761] 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[247].

Methods of TCustomRichView[216]

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[281];
- LoadText, LoadTextW[338], LoadTextFromStream, LoadTextFromStreamW[339];

- AddTab[280] – adds tabulator regardless of SpacesInTab mode.

Methods of TCustomRichViewEdit[469]

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[540], InsertTextFromFile, InsertTextFromFileW[541]
- InsertTab[540] – inserts tabulator regardless of SpacesInTab mode

See also methods for pasting[113].

4.4 Pictures

Pictures

Picture item displays a picture stored in TRVGraphic[1032] object.

Delphi VCL 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).
- TSkGraphic (Skia4Delphi[163], stores WebP and WBMP)
- TSkSvgGraphic (Skia4Delphi[163], stores SVG; include RVSkia unit in your project)

In addition, you can use third-party graphic classes[1136].

Delphi FireMonkey has the following graphic classes:

- TRVRasterGraphicFM (also accessible as TRVBitmap) (wrapper for TBitmap, stores various raster format; a list of supported format depends on the platform)
- TRVSvgImageSkiaFM (Skia4Delphi[163]) (wrapper for TSkSvgBrush, stores SVG, include fmxRVSkiaFM in your project)

Style[310] of this item type: *rvsPicture* (-3)

For hypertext pictures, see *hot-pictures*[172] item type.

Methods of TCustomRichView[216]

The following viewer-style methods add item of this type to the end of the document:

- AddPicture[278],

The following viewer-style method changes the main properties of item of this type:

- SetPictureInfo[370]

The following method returns main properties of the given item of this type:

- GetPictureInfo[318]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertPicture[534]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentPictureInfo[569],
- SetPictureInfoEd[576]

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentPictureInfo[522]

Properties

Size

This item type has the following integer properties[1067] related to size:

- *rvepImageWidth*, *rvepImageHeight* stretch the image.

Layout and appearance

This item type has the following integer properties[1067] 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.
- *rvepSmoothScaling* – turn smooth scaling on/off

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[1067] is nozero, TRichView does not free it.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount[1114]). Shared images are never deactivated.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign[1100];
- extra integer properties[1067]: *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[483]. You can forbid resizing for the specific image by assigning 0 to *rvepResizable* integer property[1067].

Resizing with mouse does not modify the image itself, it changes *rvepImageWidth* and *rvepImageHeight*.

OnItemResize[587] event occurs after resizing.

Other properties

This item type has the following specific integer properties [1067]:

- *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 [1072]:

- *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 [VCL and LCL]

RichView Format (RVF) [129]

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 [1033].

If *rvfoSavePicturesBody* is excluded from RVFOptions [254], graphic data is not saved in RVF. When loading such RVF files, OnRVFPictureNeeded [401] event occurs. You can use this feature to store images in database or files.

Export to HTML [132]

By default, all pictures are saved as JPEGs. If you want to save pictures of some graphic class without conversion, call RVGraphicHandler [1125].RegisterHTMLGraphicFormat.

For example:

- Delphi: RVGraphicHandler [1125].RegisterHTMLGraphicFormat(TMyGifImage);
- C++Builder: RVGraphicHandler [1125].RegisterHTMLGraphicFormat(__classid(TMyGifImage));

The standard PNG and GIF formats may be automatically registered by TRichView, see the topic about RVGraphicHandler [1033].

If you want to save pictures in another format, use OnHTMLSaveImage [383] or OnSaveImage2 [410] events.

RTF export

RTFOptions [252] 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[1125].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[1125].RegisterPngGraphic;
- icons (they are saved as PNG in Delphi 2009+ using TPngImage class)

RTF and DocX Import

You can set RTFReadProperties[253].IgnorePictures[463] to *True*, and pictures in RTF/DocX will not be added in RichView.

RTF/DocX 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[1136].

See Also...

- Working in 256-color display mode[241];
- Animation[124];
- Smooth image scaling[125].

4.5 Hot-pictures

"Hot-picture" is a picture[169] - hypertext link[97].

Style[310] of this item type: *rvsHotPicture* (-10)

All methods working with pictures[169] work with hot-pictures. The difference is only in methods that insert hot-picture in the document.

Methods of TCustomRichView[216]

The following viewer-style methods add item of this type to the end of the document:

- AddHotPicture[274].

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertHotPicture[528]

ConvertToHotPicture[510], ConvertToPicture[510] convert picture to hot-picture and vice versa.

4.6 Breaks

Break is a horizontal line (or rectangle) of the specified width and color.

It always occupies a full line.

Style [310] of this item type: *rvsBreak* (-1)

Methods of TCustomRichView [216]

The following viewer-style methods add item of this type to the end of the document:

- AddBreak [269]
- The following viewer-style method changes the main properties of item of this type:
- SetBreakInfo [358]

The following method returns main properties of the given item of this type:

- GetBreakInfo [296]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertBreak [523]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBreakInfo [561],
- SetBreakInfoEd [557]

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentBreakInfo [512]

Properties

Main properties of *breaks*: style, width, color.

In addition to the methods listed above, they are accessible as rveipcBreakStyle, rveipcBreakWidth, rveipcBreakColor [1119] properties.

Saving and loading

Export to HTML [132]

Breaks are exported as <hr>. Color of *break* is retained only in CSS version (HTMLSaveProperties [244] .HTMLSavingType [441] = *rvhtmlstNormal*, or *rvcsssoForceNonTextCSS* is included in HTMLSaveProperties [244].CSSOptions [438]).

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

Breaks 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[352]), saved as a sequence of LineWidth '-' characters.

Markdown

Breaks can be saved and read from Markdown. Properties of breaks cannot be stored in Markdown. Color of *breaks* loaded from Markdown is defined in MarkdownProperties[245].DefaultItemProperties[452].BreakColor[446].

4.7 Controls

You can insert any Delphi/C++Builder control in the document.

Style[310] of this item type: *rvsComponent* (-5)

Methods of TCustomRichView[216]

The following viewer-style methods add item of this type to the end of the document:

- AddControl[272].

The following viewer-style method changes the main properties of item of this type:

- SetControlInfo[361]

The following method returns main properties of the given item of this type:

- GetControlInfo[301]

See also

- FindControlItemNo[295]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertControl[526]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentControlInfo[564],
- SetControlInfoEd[560]

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentControlInfo[515]

Also

- ResizeControl[552],
- ResizeCurrentControl[552],
- AdjustControlPlacement2[497]

Events:

- OnControlAction[380] – occurs when the control is loaded from RVF[129], deleted, moved to/from undo buffer, saved to RVF.

Properties

Layout and appearance

This item type has the following integer properties[1067] 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 [1119] 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 [1067] is nozero, TRichView does not free it.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign [1100];
- extra integer properties [1067]: *rvepVShift*, *rvepVShiftAbs*.

Resizing

You can allow resizing this control in editor by assigning non-zero value to its *rvepResizable* property [1067].

You can forbid mouse resizing of all images and controls in the editor by including *rvoNoImageResize* in EditorOptions [483].

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 [355].

OnItemResize [587] event occurs after resizing.

Other properties

This item type has the following specific integer properties [1067]:

- *rvepVisible* allows to hide the control (to set its Visible property to *False*).

Note about Tags of controls

VCL and LCL: TRichView 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 [96]; you can use item's tag as you wish).

FireMonkey: TRichView does not use tags of inserted components.

Saving and Loading

RVF [129]

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 [254], control itself is not saved in RVF. When loading such RVF files, OnRVFControlNeeded [400] event occurs.

Export to text, RTF, DocX, and HTML [132]

By default, controls are not exported. Use OnSaveComponentToFile [405] 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 [887] event.

Special Width Mode

Width of controls can be set in percent:

```
var Item: TRVControlItemInfo [899];
  ...
Item := RichView.GetItem [304](ItemNo) as TRVControlItemInfo;
Item.PercentWidth := 50; // 50% of document width (minus paragraph indents)
```

The effect will be after reformatting [295].

See also...

BeginOleDrag [284]

4.8 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 [180].

Style [310] of this item type: *rvsBullet* (-6)

For hypertext *bullets*, see *hotspot* [178] item type.

Methods of TCustomRichView [216]

The following viewer-style methods add item of this type to the end of the document:

- AddBullet [270]

The following viewer-style method changes the main properties of item of this type:

- SetBulletInfo [359]

The following method returns main properties of the given item of this type:

- GetBulletInfo [297]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertBullet [524]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBulletInfo [562],
- SetBulletInfoEd [558]

The following method returns main properties of the item of this type at the position of caret:

GetCurrentBulletInfo [513]

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 [1119] property.

Layout and appearance

This item type has the following integer properties [1067] 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 [1100];
- extra integer properties [1067]: *rvepVShift*, *rvepVShiftAbs*.

Resizing

Bullets cannot be resized.

Other properties

This item type has the following specific string properties [1072]:

- *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 [129]

ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded [400] occurs allowing you to provide image-lists for *bullets* and *hotspots*.

Export to RTF, DocX and HTML [132]

Bullets are exported as pictures. If you want to write special code for exporting bullets in HTML, use OnHTMLSaveImage [383] event.

4.9 Hotspots

"Hotspot" is a *bullet* [176] (image from TImageList) with hypertext link [97].

Hotspots can change their image under the mouse pointer.

Style [310] of this item type: *rvsHotspot* (-4)

Methods of TCustomRichView [216]

The following viewer-style methods add item of this type to the end of the document:

- AddHotspot [275].

The following viewer-style method changes the main properties of item of this type:

- SetHotspotInfo [364]

The following method returns main properties of the given item of this type:

- GetHotspotInfo [303],
- GetBulletInfo [297]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertHotspot [528]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentHotspotInfo [565],
- SetHotspotInfoEd [570]

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentHotspotInfo [515],
- GetCurrentBulletInfo [513]

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 [1119] property.

Memory Usage

RichView does not own this image-list, and it will not destroy it when clearing document.

4.10 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 [196],
- **TRVTableItemInfo** [901] (see for detailed information).

Style [310] 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 [469].

Extra Properties

This item type has the following specific string properties [1072]:

- *rvespImageFileName* – value of this property can be saved in HTML file as table background image location; also accessible as table.BackgroundImageFileName [905].

Methods of TCustomRichView [216]

The following viewer-style methods add item of this type to the end of the document:

- AddItem [276] (create TRVTableItemInfo [901] object, add it using AddItem [276])

The following method returns TRVTableItemInfo [901] object

- GetItem [304]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem [531] (create TRVTableItemInfo [901].object, insert it using InsertItem [531])

The following methods return TRVTableItemInfo [901] object at the position of caret:

- GetCurrentItem [517],
- GetCurrentItemEx [517].

Resizing

Users can resize table rows and columns, if *rvtoRowSizing* and/or *rvtoColSizing* are included in table.Options [916].

Resizing changes BestWidth [955] and BestHeight [955] properties of table cells.

OnItemResize [587] event occurs after resizing.

Saving and Loading

See Export of tables [209].

4.11 List markers

This is a very special item type representing paragraph bullet or number.

Style[310] of this item type: *rvsListMarker* (-11)

See also: numbered sequences[183].

Properties of List Markers

TRVStyle[739] component has a collection of list styles (ListStyles[755]). Each list style[841] contains a collection of list levels (Levels[842] property). List level[859] 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[755];
- 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[369],
- GetListMarkerInfo[315],
- RefreshListMarkers[340].

Methods of TCustomRichViewEdit:

- ApplyListStyle[498],
- RemoveLists[552],
- ChangeListLevels[508].

Saving and Loading

RVF[129]

For list levels with ImageLists: ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded[400] occurs allowing you to provide an image-list.

HTM Export, HTMLSaveProperties[244]**.ListMarkerSavingType**[444] **=** ***rvhtmlmstNormal***

Lists are saved using <ul> and <ol> HTML tags.

Export using CSS (HTMLSaveProperties[244].HTMLSavingType[441] = *rvhtmlstNormal):*

CSS of lists are partially inserted directly in HTML (as inline attributes), partially are saved in a style sheet.

- *rvlstBullet, rvlstUnicodeBullet*[864] are exported as bullets with default markers (disc, circle or square) if possible. Otherwise, FormatString[861] is used. Font[861] is ignored;
- *rvlstDecimal, rvlstLowerAlpha,*[864] and other numbered list types are exported as lists with corresponding numbering. FormatString[861] and Font[861] are ignored;

- *rvlstPicture, rvlstImageList, rvlstImageListCounter* [864] are exported as bullets with picture. It's possible to change default image saving by OnHTMLSaveImage [383].

Indents are saved.

Export without CSS (HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstSimplified):*

- *rvlstBullet, rvlstUnicodeBullet, rvlstPicture, rvlstImageList* [864] are exported as bullets with default markers (disc, circle or square). FormatString [861], Picture [868], ImageList [863] and Font [861] are ignored;
- *rvlstDecimal, rvlstLowerAlpha,* [864] and other numbered list types are exported as lists with corresponding numbering (or as decimal, if this numbering type is not supported in HTML). FormatString [861] and Font [861] are ignored;
- *rvlstImageListCounter* [864] is saved as a decimal numbering.

Indents are not saved.

HTML Export, HTMLSaveProperties [244].ListMarkerSavingType [444] = *rvhtmlmstAsText*

Markers are saved without using special HTML keywords for lists (like <ol>, <ul>, <li>).

Font [861] and format strings [861] settings are respected.

Saving with CSS (HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstNormal):*

LeftIndent [864] and MarkerIndent [866] are retained. FirstIndent [860] are retained for non-hanging markers only (MarkerIndent>=LeftIndent).

Saving without CSS (HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstSimplified):*

All indents are ignored.

Generally, HTML files saved with this option look closer to the original.

Demo Projects

- Demos*\Assorted\ListStyles\

4.12 Labels

"Label" is a non-text item looking like text [167]. This text cannot be wrapped (always displayed on one line).

Class for this item type is **TRVLabelItemInfo** [970] (see for detailed information).

Style [310] of this item type: *rvsLabel* (-200)

The following items are inherited from labels:

- numbered sequences [183]
- footnotes [185]
- endnotes [184]
- sidenotes [186]
- text boxes [188]
- references to notes [190]

Related Properties of TRVStyle [739]

- FieldHighlightColor [746] – color for highlighting label items.

- FieldHighlightType[747] specifies when to highlight label items.

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVLabelItemInfo[970] object, add it using AddItem[276])

The following method returns TRVLabelItemInfo[970] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVLabelItemInfo[970] object, insert it using InsertItem[531])

The following method returns TRVLabelItemInfo[970] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

TextStyleNo[973] property can be modified by ApplyTextStyle[502] and ApplyStyleConversion[501] methods.

Properties

Main properties

- Text[972] – text to display;
- TextStyleNo[973] defines font and colors. This is an index in TextStyles[764] collection of the linked TRVStyle[739] component;
- ProtectTextStyleNo[972] – if *True*, editing operations cannot change TextStyleNo[973];
- MinWidth[971] – minimal width for the label item;
- Alignment[971] – horizontal alignment of text (if the item width is increased by MinWidth[971]);
- Cursor[971] – mouse cursor above this item.
- RemoveInternalLeading[972] – if *True*, the font internal leading is not included in the item height.

These properties are also accessible as *rvespcText*[1124], *rveipcProtectTextStyleNo, rveipcMinWidth, rveipcAlignment, rveipcCursor, rveipcRemoveInternalLeading*[1119] properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties[1067] 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[1100];

- extra integer properties[1067]: *rvepVShift*, *rvepVShiftAbs*.

4.13 Numbered sequences

Numbered sequence is a non-text item looking like text[167] 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**[978] (see for detailed information).

Style[310] of this item type: *rvsSequence* (-202)

The following items are inherited from numbered sequences:
- footnotes[185]
- endnotes[184]
- sidenotes[186]
- text boxes[188]

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:
- AddItem[276] (create TRVSeqItemInfo[978] object, add it using AddItem[276])

The following method returns TRVSeqItemInfo[978] object
- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:
- InsertItem[531] (create TRVSeqItemInfo[978] object, insert it using InsertItem[531])

The following method returns TRVSeqItemInfo[978] object at the position of caret:
- GetCurrentItem[517].

The following methods change properties of items as editing operations:
- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Numbered sequences have all properties of label items[181], but Text[972] is calculated automatically.

Additionally, it has properties:
- NumberType[980] – numbering type;
- SeqName[981] – name of a sequence (this item continues numbering from the previous item having the same name of a sequence);
- Reset[981] – if *True*, a numbering is not continued but reset from StartFrom[981].
- FormatString[980] – optional format string; if defined, a text to display is formatted according to this string.

These properties are also accessible as rvespcSeqName, rvespcSeqFormatString[1124], rveipcSeqStartFrom, rveipcSeqReset, rveipcSeqNumberType[1119] properties. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML[132] and text

Numbered sequences are exported as a plain text.

RTF and DocX

Numbered sequences can be exported and imported in RTF and DocX (all main properties are saved except for FormatString[980]).

You can set RTFReadProperties[253].IgnoreSequences[463] to *True*, and numbered sequences will not be imported (imported as a plain text) from RTF/DocX.

4.14 Endnotes

An endnote looks like a number (numbering type is specified in TRVStyle.EndnoteNumbering[746]). If FormatString[980] 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**[985] (see for detailed information).

Style[310] of this item type: *rvsEndnote* (-204)

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVEndnoteItemInfo[985] object, add it using AddItem[276])

The following method returns TRVEndnoteItemInfo[985] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVEndnoteItemInfo[985] object, insert it using InsertItem[531])

The following method returns TRVEndnoteItemInfo[985] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Endnotes have all properties of numbered sequences [183].

Differences:

- NumberType [980] is read-only; it always returns TRVStyle.EndnoteNumbering [746];
- SeqName [981] = '@endnote@' and must not be changed.

Saving and Loading

Export to HTML [132]

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 [183] and notes). Endnotes are saved as hyperlinks to this anchor.

RTF and DocX

Endnotes can be exported in RTF and DocX. If FormatString [980] contains a single character, it can be exported to RTF/DocX. Reset [981] and StartFrom [981] properties cannot be exported.

You can set RTFReadProperties [253].IgnoreNotes [463] to *True*, and endnotes will not be imported from RTF/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
```

4.15 Footnotes

A footnote looks like a number (numbering type is specified in TRVStyle.FootnoteNumbering [746]). If FormatString [980] 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** [988] (see for detailed information).

Style [310] of this item type: *rvsFootnote* (-203)

Methods of TCustomRichView [216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem [276] (create TRVFootnoteItemInfo [988] object, add it using AddItem [276])

The following method returns TRVFootnoteItemInfo[988] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVFootnoteItemInfo[988] object, insert it using InsertItem[531])

The following method returns TRVFootnoteItemInfo[988] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Footnotes have all properties of numbered sequences[183].

Differences:

- NumberType[980] is read-only; it always returns TRVStyle.FootnoteNumbering[746];
- SeqName[981] = '@footnote@' and must not be changed.

Saving and Loading

Export to HTML[132]

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[183] and notes). Footnotes are saved as hyperlinks to this anchor.

RTF and DocX

Footnotes can be exported in RTF and DocX. If FormatString[980] contains a single character, it can be exported to RTF/DocX.

You can set RTFReadProperties[253].IgnoreNotes[463] to *True*, and footnotes will not be imported from RTF/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 place
```

4.16 Sidenotes

A sidenote looks like a number (numbering type is specified in TRVStyle.SidenoteNumbering[746]). If FormatString[980] 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** [990] (see for detailed information).

Style [310] of this item type: *rvsSidenote* (-206)

The following items are inherited from numbered sidenotes:

- text boxes [188]

Methods of TCustomRichView [216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem [276] (create TRVSidenoteItemInfo [990] object, add it using AddItem [276])

The following method returns TRVSidenoteItemInfo [990] object

- GetItem [304]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem [531] (create TRVSidenoteItemInfo [990] object, insert it using InsertItem [531])

The following method returns TRVSidenoteItemInfo [990] object at the position of caret:

- GetCurrentItem [517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx [566];
- SetItemExtraIntPropertyExEd [572];
- SetCurrentItemExtraStrPropertyEx [567];
- SetItemExtraStrPropertyExEd [573].

Properties

Sidenotes have all properties of numbered sequences [183].

Differences:

- NumberType [980] is read-only; it always returns TRVStyle.SidenoteNumbering [746];
- SeqName [981] = '@sidenote@' and must not be changed.

Additionally, it has the following properties:

- BoxPosition [997] – properties relative to the position of a text box;
- BoxProperties [994] – 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 [1119]. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML [132]

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 [183] and notes). Sidenotes are saved as hyperlinks to this anchor.

RTF and DocX

Sidenotes can be exported in RTF and DocX. They are imported as NoteText [984] followed by a text box containing Document [984]. A bookmark named '_sidenoteN' (where *N* is an index of this sidenote) is added around NoteText, to allow saving note references [190] 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 [997].VerticalAnchor [1002];
- MS Word does not support different padding for border and background, so BoxProperties [994].Border [995].BorderOffsets [853] are not saved to RTF.

(* this information is based on tests of MS Word 2013)

Floating boxes cannot be imported from RTF (yet), but can be imported from DocX. Sidenotes are imported from DocX as text boxes [188].

In DocX, TRichView supports text boxes both in the old (VML) and the new (DrawingML) formats.

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 place
```

4.17 Text boxes

A text box item is a simplified version of a sidenote [186]. There is no visible text at the place of insertion of this item.

If rvoShowSpecialCharacters is included in RichView.Options [247], 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**[1004] (see for detailed information).

Style[310] of this item type: *rvsTextBox* (-207)

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVSidenoteItemInfo[990] object, add it using AddItem[276])

The following method returns TRVSidenoteItemInfo[990] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVSidenoteItemInfo[990] object, insert it using InsertItem[531])

The following method returns TRVSidenoteItemInfo[990] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Text box items have all properties of sidenotes[186], but almost all of them are ignored.

Only floating box-related properties are used: BoxPosition[997] and BoxProperties[994].

Saving and Loading

Export to HTML[132]

Text boxes are not exported to HTML.

RTF and DocX

Text boxes are exported to RTF and DocX and imported from DocX.

See the comments about export of sidenotes[186] 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 floa
```

4.18 References to notes

When inserted in footnote[185]'s/endnote[184]'s/sidenote[186]'s Document[984], the note reference displays the same text as the parent note.

When inserted in TCustomRichView, it displays NoteText[247].

Class for this item type is **TRVNoteReferenceItemInfo**[1006] (see for detailed information).

Style[310] of this item type: *rvsNoteReference* (-205)

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVNoteReferenceItemInfo[1006] object, add it using AddItem[276])

The following method returns TRVNoteReferenceItemInfo[1006] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVNoteReferenceItemInfo[1006] object, insert it using InsertItem[531])

The following method returns TRVNoteReferenceItemInfo[1006] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Saving and Loading

Export to HTML[132]

Note references are saved as a plain text. In future, they may be saved as hyperlinks to the parent footnote/endnote.

RTF and DocX

Note references can be exported in RTF and DocX.

You can set RTFReadProperties[253].IgnoreNotes[463] to *True*, and note references will not be imported from RTF/DocX.

4.19 Page numbers

When inserted in TRichView[216] or TRichViewEdit[469], this item is displayed as '{p}'.

When printing with TRVPrint[696], this item displays a page number where this item is located. Page numbers are counted from TRVPrint.PageNoFromNumber[881].

When drawing with TRVReportHelper[721], the default behavior is like in TRVPrint, but you can provide a value for the page number in OnGetPageNumber[728] 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**[1010] (see for detailed information).

Style[310] of this item type: *rvsPageNumber* (-208)

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVPageNumberItemInfo[1010] object, add it using AddItem[276])

The following method returns TRVPageNumberItemInfo[1010] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVPageNumberItemInfo[1010] object, insert it using InsertItem[531])

The following method returns TRVPageNumberItemInfo[1010] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Fields have all properties of label items[181].

By default, Text[972]='{p}'. You can change this value to display another code in TRichView and TRichViewEdit.

Additionally, it has property:

- NumberType[1009] – numbering type;

This property is also accessible as rveipcPageNumberType[1119] property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML[132]

Fields are saved as a plain text.

RTF and DocX

Fields can saved to RTF/DocX and loaded from RTF/DocX. You can disallow loading from RTF/DocX by assigning TRichView.RTFReadProperties[253].IgnoreFields[463]=*True*.

4.20 Page counts

When inserted in TRichView[216] or TRichViewEdit[469], this item is displayed as '{P}'.

When printing with TRVPrint[696], 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**[1010] (see for detailed information).

Style[310] of this item type: *rvsPageCount* (-209)

Methods of TCustomRichView[216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[276] (create TRVPageCountItemInfo[1011] object, add it using AddItem[276])

The following method returns TRVPageCountItemInfo[1011] object

- GetItem[304]

Methods of TCustomRichViewEdit[469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem[531] (create TRVPageCountItemInfo[1011] object, insert it using InsertItem[531])

The following method returns TRVPageCountItemInfo[1011] object at the position of caret:

- GetCurrentItem[517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

Properties

Fields have all properties of label items[181].

By default, Text[972]='{P}'. You can change this value to display another code in TRichView and TRichViewEdit.

Additionally, it has property:

- NumberType[1009] – numbering type;

This property is also accessible as rveipcPageNumberType[1119] property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML[132]

Fields are saved as a plain text.

RTF and DocX

Fields can saved to RTF/DocX and loaded from RTF/DocX. You can disallow loading from RTF/DocX by assigning TRichView.RTFReadProperties[253].IgnoreFields[463]=*True*.

Note: If RTF or DocX file is opened in Microsoft Word, this field is displayed as "?" until fields are updated.

4.21 Mathematical expressions

Items of this type display mathematical expressions.

Class for this item type is **TRVMathItemInfo** [974] (see for detailed information).

Style [310] of this item type: *rvsMathEquation* (-210)

This item requires RAD Studio XE4 or newer. This item type is not available in Lazarus and FireMonkey versions.

License

This item uses Adit Math Engine by Adit Software: ***aditmath.com***

There are two version of Adit Math Engine: free version and commercial version.

Free version

Units of the free version 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.

The free version cannot load equations from DocX and HTML, and saves them as images.

Commercial version

To use the commercial version, include RVAdvMathWrapper.pas in your project (for example, add in in "uses" of the main form unit).

Advantages: the commercial version allows saving and loading equations in HTML and DocX files.

Methods of TCustomRichView [216]

The following viewer-style method adds item of this type to the end of the document:

- AddItem [276] (create TRVMathItemInfo [974] object, add it using AddItem [276])

The following method returns TRVMathItemInfo [974] object

- GetItem [304]

Methods of TCustomRichViewEdit [469]

The following editor-style method inserts item of this type at the position of caret:

- InsertItem [531] (create TRVMathItemInfo [974] object, insert it using InsertItem [531])

The following method returns TRVMathItemInfo [974] object at the position of caret:

- GetCurrentItem [517].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[566];
- SetItemExtraIntPropertyExEd[572];
- SetCurrentItemExtraStrPropertyEx[567];
- SetItemExtraStrPropertyExEd[573].

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[1119] and rvespcText[1124] properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties[1067] 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[1119] and rvespcFontName[1124] 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[1037] object.

Saving and Loading

HTML[132]

- Free version of AME: equations are saved as images (PNG, if possible). Equations cannot be loaded.
- Commercial version of AME: equations are saved as MathML objects (can be turned off by RVMathEquationManager[1054].SaveMathMLInHTML[1021] property). Equations can be loaded from HTML.

DocX

- Free version of AME: equations are saved as images. Equations cannot be loaded.
- Commercial version of AME: equations are saved as OMML objects (can be turned off by RVMathEquationManager[1054].SaveOMMLInDocX[1021] property). Equations can be loaded from DocX.

RTF

Equations are saved as images. Equations cannot be loaded.

Export to text

Text property is saved.

4.22 Attached files

Items of this type display information about a file attached to a TRVChat[640] message.

Class for this item type is **TRVChatFileItemInfo**.

Style[310] of this item type: *rvsChatFile* (-212)

Working with this type of item is done through the TRVChat[640] methods and events. You do not need to use it directly.

4.23 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\

5 Overview of tables

Tables in TRichView

- Tables in TRichView[196]
- Addressing items in table cells (important information)[196]
- Table cells[198]
- Colors and layout of tables[199]
- Selection in a table[201]
- Cell editing[203]
- Cell merging[204]
- Table operations[205]
- Undo/redo in tables[208]
- Export of tables[209]
- Table resizing[211]
- Keys in editor[212]

Examples

- Converting table to text[212]
- Dividing table[214]

Main Classes

- TRVTableItemInfo[901] – table
- TRVTableCellData[951] – table cell

5.1 Tables in TRichView

Table is a special kind of RichView items [90], 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[901];
...
table := TRVTableItemInfo.CreateEx[928](RowCount, ColumnCount, RVData);
```

where RVData is a richview document where you wish to insert table. Usually this is richview.RVData [254] or richviewedit.RVData.

How to append table to RichView [216]

```
table.ParaNo := <Paragraph Style Index>
richview.AddItem[276](<Table name>, table)
```

where <table name> is any string associated with table, <paragraph style index> is the index in the collection of paragraph styles [757]. 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 [295] before displaying the document.

How to insert table in RichViewEdit [469]

```
richviewedit.InsertItem[531](<Table name>, table)
```

This method inserts table at 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 [180].

The table's paragraph style defines its position: Alignment [829], LeftIndent [832], RightIndent [835] (FirstIndent [831] is ignored). The paragraph style can define border [831] around the table and background [830]. All these properties are applied to the table itself, not to its cells.

5.2 Addressing items in table cells

Main Methods

RichViewEdit.**Insert*** ** methods [119] insert item(s) at 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[522] and GetPictureInfo[318])

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[489].

All methods (and properties) of editor working with current or selected item(s) can work with cell inplace editors

(for example, ApplyTextStyle[502], SetCurrentTag[570], CurTextStyleNo[482], 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[489]).

There are some methods which do not have ItemNo parameter:

- ResizeCurrentControl[552] (alternative to ResizeControl[552])
- AdjustControlPlacement2[497](Control) (alternative to AdjustControlPlacement[497](ItemNo))
- SetCurrentItemText[568](text); (alternative to SetItemTextEd[574](ItemNo, text));
- GetCurrentItemText[519] (alternative to GetItemText[310]);

Events with ItemNo Parameter

For events without RVData[1014] parameter: when events occur inside cell inplace editor, ItemNo is an index of table inside the root editor (for example OnRVMouseDown[402])

For events having both RVData and ItemNo parameter: ItemNo is an index of item in this RVData

Selection

SelectionExists[356] returns True, if something is selected inside cell, or several cells are selected in table. In this case GetSelectionBound[320] can returns -1s.

In editor, you can use TopLevelEditor.GetSelectionBounds to get information about selection in the edited cell.

Hypertext[97]

Hypertext items can be not only in main document, but in subdocuments (cells[951] or cell inplace-editor).

Use GetJumpPointLocation[312] 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 [92]

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.

5.3 Table cells

Table Cells

The main property of a table is Cells [912][<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 [951]. Each RichView control has a similar class inside (richview.RVData [254]). Cell and richview.RVData are descendants from the same class [1014] designed to store formatted RichView document.

Cells also have additional table-related properties (discussed below).

Table Rows

Tables have another property: Rows [920].

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 [196]).

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 [965].

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 [203].

5.4 Colors and layout of tables

Table Colors

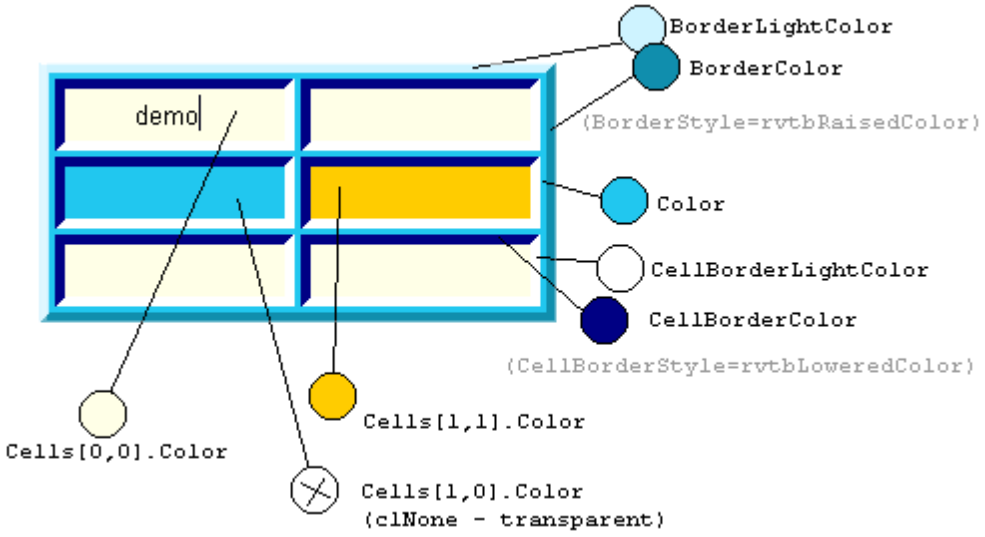


Table Colors

The main color properties of tables (Color [914], BorderColor [906], BorderLightColor [907], CellBorderColor [909], CellBorderLightColor [910]) are shown on the figure above.

There are some more color properties related to rules (lines between cells), see below [200].

Table can be transparent, if its Color [914] is set to *clNone*, or semitransparent [916].

Cells can be individually colored [957], with specified opacity [958]. You can specify default colors for groups of rows and columns [923].

Borders can be hidden, if their widths are set to 0 (see BorderWidth [909] and CellBorderWidth [910]). These are default values.

Cell borders can be 3d, like on this figure, or flat (see BorderStyle [908] and CellBorderStyle [910]).

You can hide some sides of borders of some cells (see cells' VisibleBorders [963]) and set individual color for cell border (cell.BorderColor [956] and cell.BorderLightColor [956]).

When a table with invisible border is inserted in TRichViewEdit [469], such borders are shown as dotted gray lines (see also table Options [916]).

Background Images

Tables and cells can have background images.

See properties of table: BackgroundImage [904], BackgroundProperties [905].

See properties of cell: BackgroundImage [953], BackgroundProperties [954].

Borders and Spacing

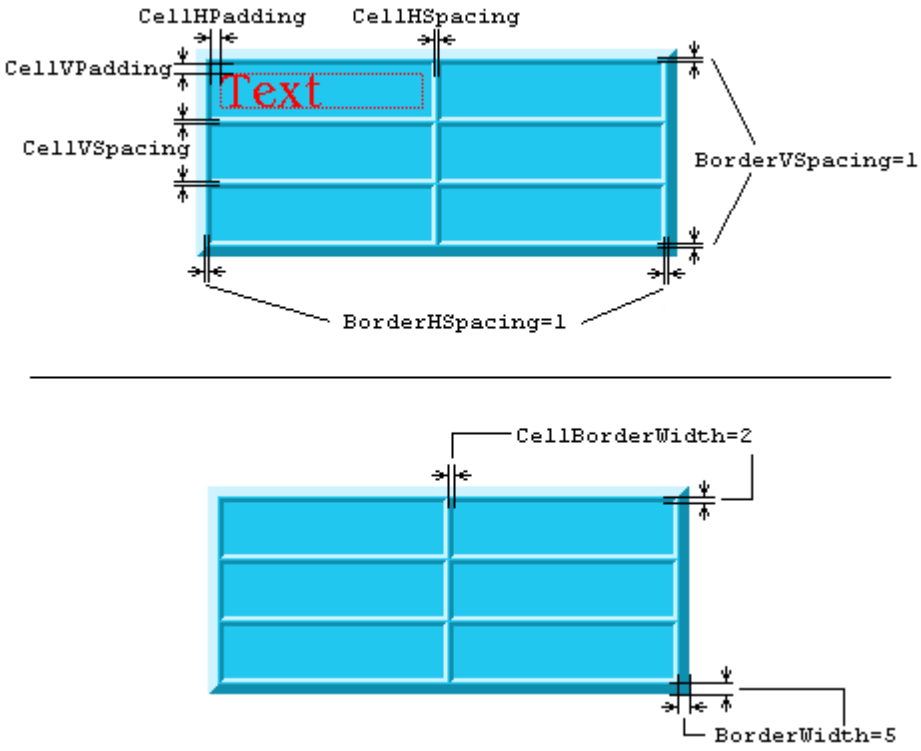


Table Borders

The figure above shows properties affecting the table layout:

- CellHSpacing [911], CellVSpacing [913],
- BorderHSpacing [907], BorderVSpacing [908],
- BorderWidth [909],
- CellBorderWidth [910],
- CellHPadding [911], CellVPadding [913].

(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.

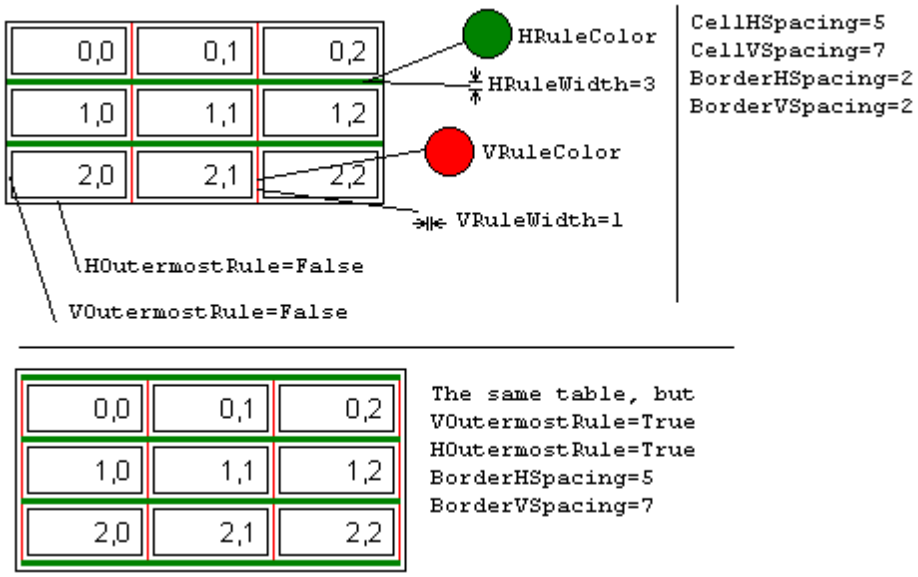


Table Rules

By default rules are invisible.

Rules are defined with properties: HRuleWidth [916], HRuleColor [915], HOutermostRule [915], VRuleWidth [922], VRuleColor [922], VOutermostRule [921].

Tip: How to make 1 pixel border

TRichView tables have HTML-style layout, so making one-pixel width borders is a trick.

Even if you set CellBorderWidth [910]=1, CellHSpacing [911]=CellVSpacing [913]=0, 1-pixel width borders of adjacent cells will be visually combined in 2-pixel width borders.

Recommended solution:

setting CellBorderWidth [910]=1, CellHSpacing [911]=CellVSpacing [913]=-1.

This example assumes RVStyle.Units [764]=*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 [963] to hide one of adjacent border sides

5.5 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[320] (and compare its results with GetItemNo[309] 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[933] or GetNormalizedSelectionBounds[933] (the latter is more convenient).

You can select some cells in table using TRVTableItemInfo.Select[939] and remove the selection using TRVTableItemInfo.Deselect[931].

You can select some rows or columns completely: TRVTableItemInfo.SelectRows[940] and SelectCols[940].

You can determine if the given cell is selected or not with the method IsCellSelected[936].

Selection Inside Cell

You can get inplace-editor and position for currently edited cell: GetEditedCell[932].

See also: TRichViewEdit TopLevelEditor[489].

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[931](0,0);
table.Cells[912][0,0].GetRVData[964].SelectAll[355];
```

Example 2 (does the same)

```
uses CRVData, CRVFData;
var RVData: TCustomRVData[1014];
...
RVData := table.Cells[0,0];
RVData := RVData.Edit;
TCustomRVFormattedData[1014](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[469] methods can be applied to the selected cells (regardless the type of selection):

- ApplyTextStyle[502]

- ApplyParaStyle [498]
- ApplyStyleConversion [501]
- ApplyParaStyleConversion [499]
- DeleteSelection [511].

5.6 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 [916])

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 [469], not in TRichView [216].

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 [965], use methods of cell's GetRVData [964] instead of methods of the cell itself.

You can initiate editing of cell from code: table.EditCell [931] (see the example in the topic about selection in tables [201]) or Cell.Edit [1016].

Position of edited cell can be returned by method GetEditedCell [932] and GetNormalizedSelectionBounds [933].

Limitations

This mechanism of cell editing adds some limitations:

You must not destroy inplace editor from inside the following events: OnKeyDown, OnKeyUp, OnKeyPress, OnRVMouseMove [403], OnStartDrag, OnDragOver.

Inplace editor can be destroyed by calling RichViewEdit.Clear [286], or by the methods placing caret/selection outside the editing cell (RichViewEdit.SetSelectionBounds [372], table.EditCell [931], 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*\Assorted\Graphics\DragImg\

5.7 Cell merging

Cells have read-only ColSpan [957] and RowSpan [962] properties, similar to html <td colspan rowspan>, 1 by default.

If Cells [912][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.

Table: 5 rows, 3 columns

ColSpan=1, **RowSpan=3**	**ColSpan=2**, RowSpan=1	
	ColSpan=1, RowSpan=1	ColSpan=1, RowSpan=1
ColSpan=1, RowSpan=1	**ColSpan=2, RowSpan=3**	
ColSpan=1, RowSpan=1		

ColSpan and RowSpan example

Example: iterating through all cells of table:

```
for r := 0 to table.RowCount[920]-1 do
 for c := 0 to table.ColCount[914]-1 do
   if table.Cells[912][r,c] <> nil then
     table.Cells[912][r,c].Color[957] := 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 [920].GetMainCell method:

```
function GetMainCell[969](ARow,ACol: Integer;
  out MRow, MCol: Integer): TRVTableCellData[951];
```

Methods for merging cells:

- MergeCells [937],

- MergeSelectedCells [937]

Inverse operation:

- UnmergeCells [947],
- UnmergeSelectedCells [947]

Splitting cells:

- SplitSelectedCellsHorizontally [946],
- SplitSelectedCellsVertically [946]

5.8 Table operations

Operations on Selection [201]

Inserting rows and columns:

- InsertColsLeft [934],
- InsertColsRight [935],
- InsertRowsAbove [936],
- InsertRowsBelow [936].

Deleting rows and columns:

- DeleteSelectedCols [930],
- DeleteSelectedRows [930],

Cell merging and unmerging [204]:

- MergeSelectedCells [938] (see also CanMergeSelectedCells [926]),
- UnmergeSelectedCells [947],
- SplitSelectedCellsHorizontally [946],
- SplitSelectedCellsVertically [946]

Other Operations

Inserting rows and columns:

- InsertCols [934],
- InsertRows [935].

Deleting rows and columns:

- DeleteCols [928],
- DeleteRows [929].

Cell merging and unmerging [204]:

- MergeCells [937] (also CanMergeCells [925])
- UnmergeCells [947],
- DeleteEmptyRows [929], DeleteEmptyCols [929] (use after cell merging).

Moving rows

- MoveRows [938].

Assigning Table Properties (As Editing Operations)

Methods for assigning table properties:

- SetTableVisibleBorders [946];

- SetBackgroundProperty[940].

Methods for assigning row properties:

- SetRowVAlign, SetRowKeepTogether, SetRowPageBreakBefore[945].

Methods for assigning cell properties:

- SetCellColor[943], SetCellBorderColor[942], SetCellBorderLightColor[943];
- SetCellVAlign[945];
- SetCellRotation[944];
- SetCellTag[944];
- SetCellBestWidth[942], SetCellBestHeight[942];
- SetCellBackgroundImage[941], SetCellBackgroundImageFileName[941], SetCellBackgroundProperty[941];
- SetCellHint[943];
- SetCellVisibleBorders[945];
- SetCellOpacity[943];
- SetCellOptions[944].

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[216], you need to call Format[295] method after to update the document view.

If these operations are performed on table inserted in TRichViewEdit[469], a special sequence of steps is required:

1. checking RichViewEdit.CanChange[506] (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[503] for table
4. performing operations
5. calling rve.EndItemModify[511]
6. calling rve.Change[508]

The example is below.

You can group several actions so that they will be undone/redone as whole, using SetUndoGroupMode[577]

In order to perform operation(s) on table, you need to get table object.You can do it using method

```
function TRichView.GetItem[304](ItemNo: Integer): TCustomRVItemInfo;
```

TCustomRVItemInfo[899] is an ancestor class for all items of RichView, including table (TRVTableItemInfo[901]). 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[310](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 at position of caret with method

```
function TRichViewEdit.GetCurrentItem[517]: 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[517](
  RequiredClass: TCustomRVItemInfoClass;
  out ItemRichViewEdit: TCustomRichViewEdit;
  out 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[899];
    table: TRVTableItemInfo[901];
    Data: Integer;
    rve: TCustomRichViewEdit[469];
    ItemNo: Integer;
begin
  if not MyRichViewEdit.CanChange[506] or
     not RichViewEdit1.GetCurrentItemEx[517](TRVTableItemInfo, rve,
       item) then
    exit;
  table := TRVTableItemInfo(item);
  ItemNo := rve.GetItemNo[309](table);
  rve.BeginItemModify[503](ItemNo, Data);
  // performing some operation, for example
  // table.InsertRowsBelow(1);
  // or table.CellPadding := 10
  rve.EndItemModify[511](ItemNo, Data);
  rve.Change[508];
end;
```

5.9 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[469].

Operations will not be undone if:

- table was created, but not inserted yet;
- table was inserted in RichView[216];
- table was inserted in RichViewEdit with UndoLimit[490]=0.

Example 1

Two operations: (inserting table and changing its color) will be undone in two steps

```
table := TRVTableItemInfo.CreateEx(928)(4,3, RichViewEdit1.RVData(254));
if RichViewEdit1.InsertItem(531)('', table) then
  table.Color(957) := 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[277], for example)

These methods must not be mixed with editor-style methods (such as InsertText[540]) 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[286] 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[490] of editor = 0 (undo mechanism turned off). But Format[295] must be called before calling any editor-style method.
- After ClearUndo[509] method. But Format[295] must be called before calling any editor-style method.

Undo of Assignment to Properties

Assigning values to cell properties (and to VAlign[968] property of rows) will not be undone.

But there are some special methods.

```
table.Cells(912)[r,c].Color(957) := clRed; // will not be undone
table.SetCellColor(943)(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[503] /EndItemModify[511].

For example

```
var item: TCustomRVItemInfo;
    table: TRVTableItemInfo;
    Data: Integer;
    rve: TCustomRichViewEdit;
    ItemNo: Integer;
begin
  if not RichViewEdit1.CanChange[506] or
     not RichViewEdit1.GetCurrentItemEx[517](TRVTableItemInfo, rve,
       item) then
    exit;
  table := TRVTableItemInfo(item);
  ItemNo := rve.GetItemNo[309](table);
  rve.BeginItemModify[503](ItemNo, Data);
  table.SetRowVAlign(rvcBottom, 1);
  table.SetCellColor(clRed,1,1);
  rve.EndItemModify[511](ItemNo, Data);
  rve.Change[508];
end;
```

Assignments to all table properties can be undone (but still the sequence BeginItemModify-operations-EndItemModify-Change is required). The only exception is VisibleBorders[921] property, use SetTableVisibleBorders[946] 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[946].

Methods for assigning row properties:

- SetRowVAlign[945].

Methods for assigning cell properties:

- SetCellColor[943], SetCellBorderColor[942], SetCellBorderLightColor[943];
- SetCellVAlign[945];
- SetCellBestWidth[942], SetCellBestHeight[942];
- SetCellBackgroundImage[941], SetCellBackgroundImageFileName[941], SetCellBackgroundProperty[941];
- SetCellHint[943];
- SetCellVisibleBorders[945].

5.10 Export of tables

Tables can be exported with document in the following formats:

- text,
- HTML,
- RTF,
- DocX

- Markdown.

Text Export

Tables have the properties:

- TextRowSeparator[921] – string to write after each row;
- TextColSeparator[920] – 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

TRichView 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[912][r,c].BestWidth[906] (specified in twips, EMU 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[906] = 0 and BestWidth[906] > 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[906] by including *rvtoRTFSaveCellPixelBestWidth* to table.Options[916]; however, this is not recommended for fixed tables.

If you include *rvtoRTFAllowAutofit* in table.Options[916], this table will be saved as fit-by-content table (but only if table.BestWidth[906]=0). For such tables, you can safely include *rvtoRTFSaveCellPixelBestWidth* to table.Options[916].

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 TRichView.HTMLSaveProperties[244].HTMLSavingType[441] = *rvhtmlstNormal*, or *rvcsssoForceNonTextCSS* in TRichView.HTMLSaveProperties[244].CSSOptions[438]). See comments for table properties.

Markdown

The core Markdown specification does not support tables.

However, there is Markdown extension for tables. It can be activated by including *rvmdeTables* in TRichView.MarkdownProperties[245].Extensions[452] (included by default).

Otherwise, content of table cells is saved as normal paragraphs.

See Also...

TRichView methods for saving and loading [127]

5.11 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 [916])
- table.BestWidth [906]
- cells BestWidth [955]

Heights are calculated basing on:

- heights of cells content (if *rvtoIgnoreContentHeight* is not in table.Options [916] and cell.IgnoreContentHeight [958] = *False*)
- cells BestHeight [955]

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 [916]);
2. if table.BestWidth [906]<>0, it defines width of the table (but table can be wider because of the rule 1)
3. if table.BestWidth [906]=0 then
    - if all columns have cells with BestWidth [955]s in TRVStyleUnits [1094], i.e. pixels or twips or EMU (and do not have cells with BestWidth [955]s in %), width of table is calculated basing on cells BestWidth [955]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 [906]=-100 (100%) (in this version TRichView 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/EMU.

Resizing with Mouse

Resizing with mouse is possible only in editor.

If you include *rvtoRowSizing* and *rvtoColSizing* in table.Options [916], users will not be able to resize it.

When user resizes table with mouse, he/she actually changes values of BestWidth [955] or BestHeight [955] properties of cells.

When resizing columns with mouse, and columns to the left and to the right do not have cells with BestWidth[955]s in %, RichViewEdit adjusts BestWidth[955]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[955]s of the column to the left.

5.12 Keys in editor

Keyboard processing

When inserted in TRichViewEdit[469], the following keys are processed by tables:

Tab moves the caret to the next cell. If pressed when the caret is in the last cell, a new row is added, and the caret is moved to the first cell of this new row (this feature can be turned off by assigning *False* to RichViewTableAutoAddRow[1117]).

Shift+**Tab** moves the caret to the previous cell.

Delete clears the selected cells.

Backspace: if all cells are selected, deletes the table; otherwise, if rows are completely selected, deletes the selected rows; otherwise, if columns are completely selected, deletes the selected columns (this feature can be turned off by assigning *False* to RichViewTableAutoAddRow[1117]).

5.13 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[120] by the user.

```
procedure ConvertTableToText(RichViewEdit: TCustomRichViewEdit[469]);
var table: TRVTableItemInfo[901];
    rv: TRichView[216];
    rve: TCustomRichViewEdit[469];
    Stream: TMemoryStream;
    ItemNo,r,c: Integer;
begin
  if not RichViewEdit.GetCurrentItemEx[517](TRVTableItemInfo[901], rve,
    TCustomRVItemInfo[899](table)) then
    exit;
  rv := TRichView[216].Create[291](nil);
  try
    rv.Visible := False;
    rv.Parent := RichViewEdit.Parent;
    rv.Style[260] := RichViewEdit.Style[260];
    rv.RVFTextStylesReadMode[258] := rvf_sIgnore;
    rv.RVFParaStylesReadMode[258] := rvf_sIgnore;
    for r := 0 to table.RowCount[920]-1 do
      for c := 0 to table.ColCount[914]-1 do
        if table.Cells[912][r,c]<>nil then
        begin
          Stream := TMemoryStream.Create;
```

```
          try
            table.Cells[912][r,c].GetRVData[964].SaveRVFToStream[352](Stream,
              False, clNone, nil, nil);
            Stream.Position := 0;
            rv.InsertRVFFromStream[323](Stream, rv.ItemCount[244])
          finally
            Stream.Free;
          end;
        end;
    // GetMyItemNo is like rve.GetItemNo[309](table), but faster
    ItemNo := table.GetMyItemNo;
    Stream := TMemoryStream.Create;
    try
      rv.SaveRVFToStream[352](Stream, False);
      Stream.Position := 0;
      rve.SetSelectionBounds[372](ItemNo, 0, ItemNo, 1);
      rve.InsertRVFFromStreamEd[536](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;
}
```

5.14 Example: dividing table

```
function SplitTable(rve: TCustomRichViewEdit[469];
  table: TRVTableItemInfo[901]; RowCountInFirstTable: Integer;
  PageBreak: Boolean): Boolean;
var r, c, mr, mc: Integer;
    newtable: TRVTableItemInfo[901];
    Stream: TMemoryStream;
begin
  Result := False;
  if RowCountInFirstTable>=table.RowCount[920] then
    exit;
  for c := 0 to table.ColCount[914]-1 do
    if table.Cells[912][RowCountInFirstTable, c] = nil then
    begin
      table.Rows[920].GetMainCell[969](RowCountInFirstTable, c, mr, mc);
      if mr<RowCountInFirstTable then
        exit;
    end;
  Stream := TMemoryStream.Create;
  table.SaveRowsToStream[939](Stream, RowCountInFirstTable,
    table.RowCount[920]-RowCountInFirstTable);
  Stream.Position := 0;
  newtable := TRVTableItemInfo[901].CreateEx(0, 0, rve.RVData);
  newtable.LoadFromStream[937](Stream);
  if PageBreak then
    newtable.PageBreakBefore := True;
  Stream.Free;
  rve.BeginUndoGroup[505](rvutModifyItem);
  rve.SetUndoGroupMode[577](True);
  try
    table.DeleteRows[929](RowCountInFirstTable,
      table.RowCount[920]-RowCountInFirstTable, True);
    // GetMyItemNo is the same as rve.GetItemNo[309](table), but faster
    rve.SetSelectionBounds[372](table.GetMyItemNo, 1, table.GetMyItemNo, 1);
    rve.InsertItem[531]('', newtable);
  finally
    rve.SetUndoGroupMode[577](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 [517].

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 [920]-**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 [120] by the user.

6 Components

Main Components

Icon	Class Name	Description
	TRVStyle [739]	Component for defining visual appearance of TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616] controls. It contains some properties which can be used by several TRichView controls.
	TRichView [216]	Component for displaying rich text documents
	TRichViewEdit [469]	Component for editing rich text documents
	TRVPrint [696]	Component for printing documents contained in TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616]
	TRVPrintPreview [635]	Component allowing to show document as it will be printed by TRVPrint
	TRVReportHelper [721]	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter [687]	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL; obsolete]
	TRVSpellChecker [728]	Component for implementing spelling checking in TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616], TSRichViewEdit, TDBSRichViewEdit controls
	TRVChat [640]	A component for implementing a chat in TRichView [216].

Data-Aware Components [VCL and LCL]

Icon	Class Name	Description
	TDBRichView 604	Data-aware version of TRichView [VCL and LCL]
	TDBRichViewEdit 616	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVDataSourceLink 685	Component allowing assigning TDataSource component to controls inserted in TRichView 216, TRichViewEdit 616, TDBRichView 604, TDBRichViewEdit 616, TSRichViewEdit, TDBSRichViewEdit controls [VCL and LCL]

6.1 Visual Components

TRichView Components 215 | Visual Components

Icon	Class Name	Description
	TRichView 216	Component for displaying rich text documents
	TRichViewEdit 469	Component for editing rich text documents
	TDBRichView 604	Data-aware version of TRichView
	TDBRichViewEdit 616	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVPrintPreview 635	Component allowing to show a document as it will be printed by TRVPrint [VCL and LCL]

6.1.1 TRichView

TRichView is a control for displaying documents with pictures, tables, Delphi controls, hyperlinks, footnotes and endnotes, etc.

Unit [VCL/FMX] RichView / fmxRichView.

Syntax

```
TCustomRichView = class(TRVScroller 869)
TRichView = class(TCustomRichView)
```

TRichView does not display its content if it is not linked to TRVStyle component via Style 260 property.

The most important property settings can be done in the component editor 224 for TRichView. In Delphi (C++Builder) IDE, right click the RichView object on the form, choose "Settings" in the context menu.

Hierarchy

Hierarchy (VCL and LCL)

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRVControl
TRVScroller 869
TCustomRichView

Hierarchy (FireMonkey)

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBox
TCustomRVControl
TRVScroller 869
TCustomRichView

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

- General 217
- Background 218
- Appending Items 218
- Hypertext 218
- Style templates 218
- Scrolling, margins, sizes 219
- Selection and Clipboard 219
- Saving and loading as RVF 220
- Other methods for saving and loading 220
- LiveBindings and DB 221
- Information about items 222
- Modifying items 222
- Checkpoints 222
- Mouse events 223
- Live spelling check 223
- Picture animation 223
- Working in palette mode 223
- Custom Drawing 224
- Other properties, methods and events 224

General properties

- Style 260 – link to RVStyle component defining a lot of properties which can be used in several RichViews;
- Options 247 – some important options;
- ItemCount 244 – number of items in the document.

Properties defining background

- BackgroundPicture [229];
- BackgroundProperties [229];
- Color [233].

Appending items (Add*** methods)

Main methods for appending items:

- AddNL [277] adds one text item;
- AddBreak [269] adds break (horizontal line);
- AddBullet [270] adds bullet (image from ImageList);
- AddHotspot [275] adds hotspot (image from ImageList - hypertext link);
- AddPicture [278] adds picture;
- AddHotPicture [274] adds picture - hypertext link;
- AddControl [272] adds Delphi/C++Builder control;
- AddTab [280] adds tabulator;
- AddItem [276] – general method for appending items (usually used for tables [196]);
- also AddCheckpoint [271] – adds checkpoint which will be associated with next added item.

Additional methods for appending items:

- Add [277] adds one text item in last paragraph;
- AddFmt [273] adds one formatted string assembled from a Pascal format string and an array arguments;
- AddTextNL, -A -W [281] add several text items from text, separated by CR+LF characters;

Also

- AppendFrom [282] copies content from another RichView;
- InsertRVFFromStream [323],
- AppendRVFFromStream [283].

See also

- Clear [286] clears document;
- SetAddParagraphMode [357];
- Format [295], FormatTail [296] format the document;
- Reformat [339] reformats the document.

Hypertext

- event OnJump [390] occurs when user clicks hyperlink;
- event OnRVMouseMove [403] occurs when mouse pointer moves over hyperlink;
- events OnWriteHyperlink [418] and OnReadHyperlink [395] occur when document is saved or loaded as RTF, DocX or HTML.
- property FirstJumpNo [243] – use to set hypertext id for the first hypertext item;
- GetJumpPointLocation [312] should be used instead of GetJumpPointItemNo in applications working with tables [196];
- GetJumpPointY [313] returns Y coordinate of item with the specified hypertext id.

Style templates

Properties:

- UseStyleTemplates[263] allows or disallows using Style[260].StyleTemplates[762].
- StyleTemplateInsertMode[260] specifies how style templates are used when inserting RTF or RVF content.

Events:

- OnStyleTemplatesChange[417] occurs then the collection of style templates has been modified.

Scrolling, margins and sizes

Properties:

- LeftMargin[245], RightMargin[251], TopMargin[262], BottomMargin[231] – margins;
- MaxTextWidth[246] – maximal width for word wrapping;
- MinTextWidth[246] – minimal width for word wrapping (minimal width of scrollable area, not including margins);
- DocumentHeight[239] – height of document (height of scrollable area);
- VAlign[264] – vertical alignment of document;
- VScrollVisible[874] – hides/shows vertical scrollbar;
- HScrollVisible[872] – hides/shows horizontal scrollbar;
- VSmallStep[264] – number of pixels in one step of vertical scrollbar;
- FirstItemVisible[243] – the first visible item;
- LastItemVisible[244] – the last visible item;
- MaxLength[246] – maximal count of characters per line;
- WordWrap[265] allows to disable word wrapping;
- Tracking[873] – whether RichView repaints while its scrollbar thumb tab is dragged with the mouse.

Methods:

- ClientToDocument[286] converts client coordinates to document coordinates

Events:

- OnVScrolled[877] occurs when document is scrolled vertically
- OnHScrolled[877] occurs when document is scrolled horizontally

Also:

- property CPEventKind[233]
- event OnCheckPointVisible[379]
- method GetItemCoords[306]

Methods for working with selection and the Clipboard

- SelectionExists[356] returns "is something selected?"l

- Copy[288] copies selection to the Clipboard in all available formats (except for FireMonkey for Android and iOS, where the most appropriate format is copied);
- CopyDef[288] copies selection to the Clipboard using formats specified in Options[247] (except for FireMonkey for Android and iOS, where the most appropriate format is copied);
- CopyImage[289] copies selected image to the Clipboard;
- CopyText[290] copies selection to the Clipboard as Unicode text;
- CopyRVF[290] copies selection as RVF[113] to the Clipboard;
- CopyRTF[290] copies selection as RTF (Rich Text Format);
- CopyMarkdown[289] copies selection the Clipboard as Unicode text in Markdown format.

- GetSelectedImage[319] returns selected image
- GetSelText[321] returns selection as text
- GetSelTextW[321] returns selection as Unicode string;

- GetOffsBeforeItem[317], GetOffsAfterItem[316] return offset before/after the given item; useful with SetSelectionBounds[372];

- SelectAll[355] selects the whole document for copying to the Clipboard;
- SetSelectionBounds[372] selects the specified items;
- SelectWordAt[356] selects the word at (X,Y);
- SelectControl[355] selects a visual control;
- Deselect[294] clears selection (not contents, but bounds);

- GetSelectionBounds[320] returns bounds of selection;
- event OnSelect[416] occurs when selection bounds are changed;
- event OnCopy[381] allows to copy selection to the Clipboard in custom formats;
- property SelectionHandlesVisible[260] shows or hides selection handles for a touch screen.

Properties, methods and events for loading and saving RVF

Properties:

- RVFOptions[254] – options for saving and loading RVF;
- RVFWarnings[258] describes warnings or errors occurred during the last reading of RVF.

Events:

- OnRVFControlNeeded[400] requests inserted control when reading RVF, if RVF was saved without controls' "bodies";
- OnRVFImageListNeeded[400] requests ImageList when reading bullet[176] or hotspot[178] from RVF;
- OnRVFPictureNeeded[401] requests image when reading RVF, if RVF was saved without images' "bodies".

Methods:

- SaveRVF[351] saves document or selection as RVF to the file;
- SaveRVFToStream[352] saves document or selection as RVF to the stream;
- LoadRVF[336] loads RVF from the file;
- LoadRVFFromStream[337] loads RVF from the stream;
- CopyRVF[290] copies selection as RVF to the Clipboard;
- InsertRVFFromStream[323] inserts RVF from the stream;
- AppendRVFFromStream[283] appends RVF from the stream using the specified paragraph style for the first RVF item.

Also:

- methods Copy[288], CopyDef[288].

Other saving and loading

Methods:

- SetHeader[363], SetFooter[362] specify documents for using as headers and footers; they are used to load and save headers and footers to files (RVF, RTF, DocX)

- LoadFromStream, LoadFromStreamEx [329] load document in RVF, RTF, HTML, DocX, Markdown or text format (format detection) from a stream;
- LoadFromFile, LoadFromFileEx [328] load document in RVF, RTF, HTML, DocX, Markdown or text format (format detection) from a file;
- LoadHTML [330], LoadHTMLFromStream [332] appends a file or a stream in HTML format;
- LoadText [338] appends text form a file;
- LoadTextW [338] appends text from a file in Unicode (UTF-16) encoding;
- LoadRTF [334], LoadRTFFromStream [335] appends a file or a stream in RTF (Rich Text Format);
- LoadMarkdown [332], LoadMarkdownFromStream [333] appends a file or a stream in ***Markdown*** format;
- SaveHTML [342], SaveHTMLToStream [346] save the document in an HTML file/stream;
- SaveText [352], SaveTextToStream [353] save the document in a file/stream as ANSI text;
- SaveTextW [352], SaveTextToStreamW [353] save the document in a file/stream as a Unicode text;
- SaveDocX [341], SaveDocXToStream [342] save the document (or selection) in a file/stream as DocX (Microsoft Word format);
- SaveRTF [350], SaveRTFToStream [351] save the document (or selection) in a file/stream as RTF.
- SaveMarkdown [348], SaveMarkdownToStream [349] save the document (or selection) in a file/stream as ***Markdown***.

Properties:

- RTFOptions [252] – options for DocX and RTF export.
- RTFReadProperties [253] – properties for DocX and RTF import.
- HTMLReadProperties [243] – properties for HTML import.
- HTMLSaveProperties [244] – properties for HTML export.
- MarkdownProperties [245] – properties for ***Markdown*** import and export.

Events:

- OnSaveComponentToFile [405] allows to save inserted controls in DocX, RTF, Markdown, HTML, and text files;
- OnWriteHyperlink [418], OnReadHyperlink [395] allow to customize saving/loading of hypertext links in HTML, Markdown, RTF and DocX files;
- OnWriteObjectProperties [420] allows saving additional properties of non-text objects in HTML, RTF and DocX files;
- OnReadMergeField [398] allow importing merge fields from RTF;
- OnHTMLSaveImage [383], OnSaveImage2 [410] allow to save images with HTML in custom formats and with custom file names;
- OnSaveItemToFile [411] allows to save any item in RTF, DocX, Markdown, HTML or text files in your own format;
- OnImportPicture [386] allows custom processing of external images when importing RTF, DocX, Markdown, HTML;
- OnImportFile [385] allows custom processing of external files when importing HTML;
- OnAssignImageFileName [379] allows changing file names assigned to inserted images when importing RTF, DocX or Markdown.

LiveBindings and Databases

Properties:

- Document [238] allows linking this TRichView control to TBlobField field of DB (in Delphi XE2+)

Events:

- OnNewDocument[391] occurs when creating or before loading a document from a DB field
- OnLoadDocument[391] occurs after loading a document from a DB field
- OnLoadCustomFormat[390] allows loading documents from a DB field in custom formats

Information about items (Get***Info methods)

- GetItemStyle[310] returns a type of item;

- GetTextInfo[321] – information about text item;
- GetBreakInfo[296] – about "break";
- GetBulletInfo[297] – about "bullet";
- GetHotspotInfo[303] – about "hotspot";
- GetPictureInfo[318] – about picture or "hot-picture";
- GetControlInfo[301] – about inserted control;
- GetItemExtraIntProperty[Ex][307], GetItemExtraStrProperty[Ex][308] return value of extra item property;
- GetItem[304] – general method for obtaining items (usually used for tables[196]);

- GetItemCheckpoint[305] – about item's "checkpoint";
- GetItemVAlign[311] – about item's position relative to line;
- GetItemTag[310] – about item's "tag";
- GetItemText[310] – about text/name of item;
- IsParaStart[325] returns "is this item a first item in paragraph?";
- IsFromNewLine[325] returns "is this item a first item in paragraph?";
- GetItemPara[309] returns an index of paragraph containing a specified item;
- GetItemNo[309] returns item index for object representing item (usually used for tables[196]);
- GetItemCoords[306] returns coordinates of item relative to document area, GetItemClientCoords[306] returns coordinates of item relative to client area of RichView window;
- properties PageBreaksBeforeItems[250], ClearLeft[232], ClearRight[232].

Modifying items (Set***Info methods)

- SetBreakInfo[358] changes *break*[173];
- SetBulletInfo[359] changes *bullet*[176];
- SetHotspotInfo[364] changes *hotspot*[178];
- SetPictureInfo[370] changes picture[169] or *hot-picture*[172];
- SetControlInfo[361] changes inserted control[174];
  also
- SetItemTag[367] changes item's "tag";
- SetItemVAlign[369] – change the item position relative to the line;
- SetItemText[368] changes text/name of item;
- SetItemExtraIntProperty[Ex][366], SetItemExtraStrProperty[Ex][367] sets value of extra item property;
- SetCheckpointInfo[360] sets or changes item's "checkpoint";
- RemoveCheckpoint[340] removes item's "checkpoint";
- see also properties PageBreaksBeforeItems[250], ClearLeft[232], ClearRight[232].

Working with *checkpoints*

Obtaining checkpoint data:

- GetFirstCheckpoint[302], GetNextCheckpoint[315] allow to receive full list of *checkpoints*;
- GetLastCheckpoint[313], GetPrevCheckpoint[319] – the same in backward direction;
- GetCheckPointByNo[298] returns a *checkpoint* with the specified index;
- GetItemCheckpoint[305] returns the item's *checkpoint*;
- event OnCheckPointVisible[379] occurs when *checkpoint* (or "section", marked by this *checkpoint*) becomes visible as a result of vertical scrolling;
- FindCheckPointByName[294] searches for *checkpoint* with the specified name;
- FindCheckPointByTag[295] searches for *checkpoint* with the specified *tag*.

Extracting properties from *checkpoint* data:

- GetCheckpointInfo[299] returns main properties of the *checkpoint*;
- GetCheckpointItemNo[299] returns an index of item which owns the *checkpoint*;
- GetCheckpointNo[299] returns an index of the *checkpoint*;
- GetCheckpointYEx[301], GetCheckpointXY[300] returns coordinates of the *checkpoint*.

Also:

- GetCheckPointY[300] returns Y coordinate of *checkpoint* by its index.

See also methods for obtaining information about items[222] and for modifying document[222].

Mouse events

- OnRVDblClick[399] occurs on double click;
- OnRVMouseDown[402] occurs when user presses mouse button;
- OnRVMouseUp[403] occurs when user releases mouse button;
- OnRVRightClick[404] occurs on right click.
- OnItemHint[388] allows customizing hints displayed when the mouse pointer moves above an item.
- OnGetItemCursor[382] allows customizing cursors displayed when the mouse pointer moves above an item.

See also hypertext events[218].

Live spelling check

- StartLiveSpelling[374] starts a live spelling thread;
- ClearLiveSpellingResults[286] clears all live spelling underlines and stops the thread;
- LiveSpellingValidateWord[326] removes live spelling underlines from the specified word;
- OnSpellingCheck[416] event – the place to check word spelling.

Picture animation

Properties:

- AnimationMode[228] specifies when image animation[124] starts.

Methods:

- StartAnimation[374] starts all animations;
- StopAnimation[375] stops all animations;
- ResetAnimation[341] rewinds all animations to the first frame.

Working in palette mode

- property DoInPaletteMode[241];
- method UpdatePaletteInfo[375].

Custom drawing

Events:

- OnPaint [392] – painting on top of content;
- OnAfterDrawImage [378] – painting on top of images.

Others

Properties:

- RVData [254] – object representing document;
- Cursor [234] – cursor;
- Delimiters [235] – characters not belonging to words;
- TabNavigation [262] sets mode for navigation through hypertext links and controls;
- DocObjects [235] – additional object information for saving in RVF [129];
- DocProperties [238] – additional text information for saving in RVF [129];
- DocParameters [237] – store document page parameters;
- NoteText [247] – text for references to parent notes;
- UseVCLThemes [264] [VCL] allows using colors of VCL themes (in Delphi XE2+);
- UseFMXThemes [263] [FMX] allows using text color of FMX style
- DocumentPixelsPerInch [239] [VCL,LCL] / ZoomPercent [265] [FMX] allows zooming.

Methods:

- DeleteUnusedStyles [293] deletes unused text, paragraph and list styles from TRVStyle;
- MarkStylesInUse [339], DeleteMarkedStyles [292] – advanced methods for deleting unused styles;
- SearchText, -A, -W [354] search for the text string; OnTextFound [418] event occurs when a text if found;
- StoreSearchResult [375] allows storing a position of a search result (to continue search from this position);
- GetWordAt [322] returns a word at the specified point;
- DeleteItems [291], DeleteSection [293] delete the specified range of items from document;
- DeleteParas [293] deletes the specified range of paragraphs, updates the document;
- FindControlItemNo [295] returns index of item which owns the specified inserted control [174];
- GetFocusedItem [302] returns the item in the focus;
- GetLineNo [314] returns line number in the specified position;
- BeginOleDrag [284] starts drag&drop.

Events:

- OnItemAction [387];
- OnControlAction [380].

6.1.1.1 TRichView: component editor

Component editor for TCustomRichView (and all inherited components: TRichView [216], TRichViewEdit [469], TDBRichView [604], TDBRichViewEdit [616]).

To display the component editor dialog in Delphi (C++Builder) IDE, right click TRichView 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 [258], RVFParaStylesReadMode [258], RVFOptions [254], RTFReadProperties.TextStyleMode [466], .ParaStyleMode [464], UseStyleTemplates [263]. 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 [764], paragraph [757] and list [755] styles of the linked [260] TRVStyle [739] component;
- does not save these styles in RVF [129] files or streams.

"Allow adding styles dynamically" mode

In this mode, the component:

- adds new items in the collection of text [764], paragraph [757] and list [755] styles, if necessary; new styles can be added as a result of RVF, RTF, DocX loading;
- saves these styles in RVF [129] files and streams.

If you want to display RTF and DocX files close to the original, or use commands for adding styles (for example, inside OnStyleConversion [595] or OnParaStyleConversion [592], 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 [263]=*True*.

Additional information about these modes can be found in the topic "Styles and style templates" [103].

RVF [129]

Defines options for saving and loading RVF (RichView Format) [129] files and strings. These settings change RVFOptions [254].

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.

6.1.1.2 Properties

Derived from TCustomRichView

- AnimationMode 228
- BackgroundPicture 229
- BackgroundProperties 229
- BiDiMode 230
- BottomMargin 231
- ClearLeft 232
- ClearRight 232
- Color 233
- CPEventKind 233
- Cursor 234
- DarkMode 234
- Delimiters 235
- DocObjects 235
- DocParameters 237
- DocProperties 238
- Document 238
- DocumentHeight 239
- DocumentPixelsPerInch 239 [VCL,LCL]
- DoInPaletteMode 241
- FirstItemVisible 243
- FirstJumpNo 243
- HTMLReadProperties 243
- HTMLSaveProperties 244
- ItemCount 244
- LastItemVisible 244
- LeftMargin 245
- MarkdownProperties 245
- MaxLength 246
- MaxTextWidth 246
- MinTextWidth 246
- NoteText 247
- Options 247
- PageBreaksBeforeItems 250
- RightMargin 251
- RTFOptions 252
- RTFReadProperties 253
- RVData 254
- RVFOptions 254
- RVFParaStylesReadMode 258
- RVFTextStylesReadMode 258
- RVFWarnings 258
- SingleClick 260 (deprecated)
- SelectionHandlesVisible 260 (D2010+)
- **Style** 260

- StyleTemplateInsertMode 260
- TabNavigation 262
- TopMargin 262
- UseFMXThemes 264 [FMX]
- UseStyleTemplates 263
- UseVCLThemes 264 [VCL]
- VAlign 264
- VSmallStep 264 [VCL,LCL]
- WordWrap 265
- ZoomPercent 265 [FMX]

Derived from TRVScroller 869

- BorderStyle 871 [VCL]
- HScrollMax 871
- HScrollPos 872
- HScrollVisible 872 [VCL,LCL]
- InplaceEditor 872
- NoVScroll 873
- Tracking 873
- UseXPThemes 873 [VCL,LCL]
- VScrollMax 874
- VScrollPos 874
- VScrollVisible 874 [VCL,LCL]
- WheelStep 875 [VCL,LCL]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- Align
- Anchors
- AutoHide [FMX]
- Constraints
- Ctl3D [VCL]
- DisableMouseWheel [FMX]
- DragMode
- EnableDragHighlight [FMX]
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- Left [VCL, LCL]
- Locked [FMX]
- Margins [FMX]
- Name
- Opacity [FMX]
- Padding [FMX]

- ParentCtl3D [VCL]
- ParentShowHint
- PopupMenu
- Position [FMX]
- RotationAngle [FMX]
- RotationCenter [FMX]
- Scale [FMX]
- ShowHint
- ShowScrollBars [FMX]
- ShowSizeGrip [FMX]
- Size [FMX]
- StyleLookup [FMX]
- TabOrder
- TabStop
- Touch (D2010+) [VCL,FMX]
- TouchTargetExpansion [FMX]
- Tag
- Top [VCL, LCL]
- Visible
- Width

6.1.1.2.1 TRichView.AllowSelection

```
property AllowSelection: Boolean;
```

This property still works, but obsolete, please use *rvoAllowSelection* in Options [247]

6.1.1.2.2 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 [374] and stops when calling StopAnimation [375].
rvaniOnFormat	Animation starts when calling Format [295] and stops when calling Clear [286]. StartAnimation [374] and StopAnimation [375] work as well.

Default value:

rvaniManualStart

See also:

- Animation in TRichView [124];
- Pictures [169];
- *Hot-Pictures* [172].

6.1.1.2.3 TRichView.BackgroundBitmap

Background image shown below the document

```
property BackgroundBitmap: TBitmap;
```

Deprecated, use BackgroundPicture [229] instead.

6.1.1.2.4 TRichView.BackgroundPicture

Background image for the whole document.

```
property BackgroundPicture: TRVPicture [1037];
```

This picture is shown if BackgroundProperties [229].Visible [1025] = *True*.

See also

- BackgroundProperties [229]

6.1.1.2.5 TRichView.BackgroundProperties

Defines how BackgroundPicture [229] is displayed

```
property BackgroundProperties: TRVBackgroundProperties [1021];
```

In TRichViewEdit [469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd [555].

See also properties:

- BackgroundPicture [229];
- Color [233].

See also:

- Smooth image scaling [125].

6.1.1.2.6 TRichView.BackgroundStyle

Defines how BackgroundPicture [229] is displayed

```
type
  TBackgroundStyle = (bsNoBitmap, bsStretched,
    bsTiled, bsTiledAndScrolled, bsCentered,
    bsTopLeft, bsTopRight,
    bsBottomLeft, bsBottomRight, bsOther);

property BackgroundStyle : TBackgroundStyle;
```

Deprecated, use BackgroundProperties [229] instead.

Note: TBackgroundStyle is declared in RVScroll unit.

Style	Meaning
bsNoBitmap	Colored background (picture is not used)
bsStretched	Picture is stretched to fit TRichView window
bsTiled	Repeats picture until the entire TRichView window is covered
bsTiledAndScrolled	Repeats picture until the entire scrollable area is covered (tiles are scrolled together with content)
bsCentered	Centers picture in TRichView window
bsTopLeft,	Picture is in the top left corner
bsTopRight	Picture is in the top right corner
bsBottomLeft	Picture is in the bottom left corner
bsBottomRight	Picture is in the bottom right corner
bsOther	The value specified in BackgroundProperties does not have an analog in TBackgroundStyle

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

See also:

- TRVItemBackgroundType[1080] type;
- Smooth image scaling[125].

6.1.1.2.7 TRichView.BiDiMode

Specifies the default bidirectional mode for the control.

```
property BiDiMode: TRVBiDiMode[1058];
```

(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

Value	Meaning
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[831] and BiDiMode of text style[808].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

The results depends on the value of Style[260].TextEngine[763]:

- 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[137].

Default value:

rvbdUnspecified

6.1.1.2.8 TRichView.BottomMargin

Bottom margin of document

```
property BottomMargin: TRVPixelLength[1085];
```

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style[260].UnitsPerInch[765] of an inch. When drawing or a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

See also properties:

- LeftMargin[245];
- RightMargin[251];
- TopMargin[262].

See also properties of TRVPrint:

- Margins[705].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[110]

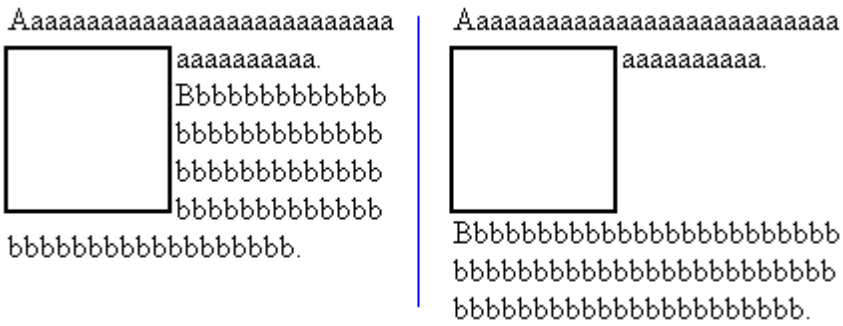
6.1.1.2.9 TRichView.ClearLeft

Specifies whether a text flow around left-aligned[1100] 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[1100] 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[325](ItemNo)) or a new line inside a paragraph (IsFromNewLine[325](ItemNo)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using Format[295]).

In editor you can change value of this property as an editing operation, using ClearTextFlow[509] method.

Note: if special characters are shown (i.e. *rvoShowSpecialCharacters* is included in Options[247]), 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[232],
- PageBreaksBeforeItems[250],
- TRVVAlign[1100] type (*rvvaLeft*).

6.1.1.2.10 TRichView.ClearRight

Specifies whether a text flow around right-aligned[1100] 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[1100] object.

This item must start a new paragraph (IsParaStart[325](ItemNo)) or a new line inside a paragraph (IsFromNewLine[325](ItemNo)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using Format[295]).

In editor you can change value of this property as an editing operation, using ClearTextFlow[509] method.

Note: if special characters are shown (i.e. *rvoShowSpecialCharacters* is included in Options[247]), 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[232],
- PageBreaksBeforeItems[250],
- TRVVAlign[1100] type (*rvvaRight*).

6.1.1.2.11 TRichView.Color

Background color

```
property Color: TRVColor[1063];
```

This property allows to override the default background color specified in RichView.Style[260].Color[744].

TRichView paints background with this color, if it is not equal to *rvclNone*[1105], otherwise TRichView uses the default color.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Default value:

rvclNone[1105]

See also properties:

- BackgroundProperties[229];
- BackgroundPicture[229].

See also properties of TRVStyle:

- Color[744].

6.1.1.2.12 TRichView.CPEventKind

Specifies the mode of generating OnCheckpointVisible[379] event.

```
type
  TCPEventKind =
   (cpeNone, cpeAsSectionStart, cpeWhenVisible);

property CPEventKind: TCPEventKind;
```

Note: TCPEventKind is declared in RVScroll unit.

Value	Meaning
cpeNone	OnCheckpointVisible[379] does not occur

cpeAsSectionStart	OnCheckpointVisible[379] 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[379] 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[379].

See also:

- *Checkpoints*[92] for details.

6.1.1.2.13 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[469].

See also:

- RichViews' cursors[147].

6.1.1.2.14 TRichView.DarkMode

Inverts luminance of all colors

```
property DarkMode: Boolean;
```

(introduced in version 23)

If *True*, luminance of all colors is inverted. Dark colors become light colors, black becomes white, white becomes black.

Additionally, in FireMonkey, if both Color[233] and Style[260].Color[744] are equal to *TAlphaColorRec.Null*, black background is drawn.

This property affects text, lines and backgrounds. It does not affect inserted images.

See also:

- UseVCLThemes[264]
- UseFMXThemes[263]
- TCustomRVPrint[877].DarkMode[878]

6.1.1.2.15 TRichView.Delimiters

Set of delimiters for specifying a "word" in document

```
property Delimiters: TRVUnicodeString[1099];
```

(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[356];
- GetWordAt[322];
- SearchText[354].

See also events:

- OnRVDblClick[399];
- OnRVRightClick[404].

See also methods of TRichViewEdit:

- SelectCurrentWord[555].

See also:

- Live spelling check[138].

6.1.1.2.16 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 [254], these objects are saved in RVF.

If *rvfoLoadDocObjects* is in RVFOptions [254], methods for loading RVF load these objects. Methods for inserting RVF ignore these objects.

Method TRichView.Clear [286] clears **DocObjects**.

There is one more property allowing to save custom data in RVF: DocProperties [238].

Example 1

TRVMathDocObject [1037] defines default properties for items displaying mathematical expressions [193].

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 [254].

How to load from RVF: include *rvfoLoadDocObjects* in RVFOptions [254].

See also methods for saving RVF (they can save this property):

- SaveRVF [351];
- SaveRVFToStream [352].

See also methods for loading RVF (they can load this property):

- LoadRVF[336];
- LoadRVFFromStream[337].

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd[537];
- InsertRVFFromStreamEd[536];
- PasteRVF[547].

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream[323];
- AppendRVFFromStream[283].

See also properties:

- RVFOptions[254].

See also:

- Saving and loading RichView documents[127];
- RVF[129].

6.1.1.2.17 TRichView.DocParameters

Page properties.

```
property DocParameters: TRVDocParameters[422];
```

(introduced in version 10)

Subproperties of this property are not used by TRichView itself, but can be written and read in RVF[129], RTF, DocX, HTML.

Besides, they can be assigned to TRVPrint[696] using the method AssignDocParameters[708].

RVF

DocParameters are stored with DocProperties[238].

How to save to RVF: include *rvfoSaveDocProperties* in RVFOptions[254].

How to load from RVF: include *rvfoLoadDocProperties* in RVFOptions[254].

RTF and DocX

How to save to RTF/DocX: include *rvrtfSaveDocParameters* in RTFOptions[252].

How to load from RTF/DocX: set RTFReadProperties[253].ReadDocParameters[465] = *True*.

HTML

Title[427] and Comments[424] can be stored in HTML.

How to save to HTML: Title and Comments are always saved in HTML.

How to load from RTF: set HTMLReadProperties[243].ReadDocParameters[435] = *True*.

See also events:

- OnSaveRTFExtra[414];
- OnSaveDocXExtra[407];
- OnSaveHTMLExtra[409].

6.1.1.2.18 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 [254], these strings are saved in RVF.

If *rvfoLoadDocProperties* is in RVFOptions [254], 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 [235].

Method TRichView.Clear [286] clears DocProperties.

See also methods for saving RVF (they can save this property):

- SaveRVF [351];
- SaveRVFToStream [352].

See also methods for loading RVF (they can load this property):

- LoadRVF [336];
- LoadRVFFromStream [337].

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd [537];
- InsertRVFFromStreamEd [536];
- PasteRVF [547].

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream [323];
- AppendRVFFromStream [283].

See also properties:

- RVFOptions [254].

See also:

- Saving and loading RichView documents [127];
- RVF [129].

6.1.1.2.19 TRichView.Document

Allows linking this TRichView component to a database field (Delphi XE2+)

```
property Document: TRVDocumentProperty [429];
```

(introduced in v21)

LiveBindings

You can use LiveBindings to link this property to a dataset field of TBlobField type.

This is an alternative way of using TRichView components to view and edit database fields, which works both in VCL and FireMonkey frameworks. Another ways is using TDBRichView[604] and TDBRichViewEdit[616] components (available in VCL and LCL frameworks).

This property contains sub-properties controlling this linking.

Important note: in order to link **Document** to TBlobField, DBRV unit (or fmxDBRV unit for FireMonkey) must be included in the project (add it in "uses").

Copying a document

Another use of this property is copying a document between TRichView controls.

The code

```
RichView1.Document := RichView2.Document;
```

copies a document from RichView2 to RichView1.

Internally, this assignment is implemented as copying via a temporary memory stream (in RVF format[129]). After copying, the destination TRichView control is formatted (unless it is TRVReportHelper's RichView).

See also events:

- OnNewDocument[391]
- OnLoadDocument[391]
- OnLoadCustomFormat[390]

See also events of TRichViewEdit[469]:

- OnSaveCustomFormat[595]

6.1.1.2.20 TRichView.DocumentHeight

Height of document (height of scrollable area)

```
property DocumentHeight: TRVCoord[239];
```

Document must be formatted to obtain a correct value.

Example [VCL]:

```
if MyRichView.DocumentHeight>Screen.Height then
  MyRichView.Height := Screen.Height
else
  MyRichView.Height := MyRichView.DocumentHeight;
```

See also:

- Scrolling[110].

6.1.1.2.21 TRichView.DocumentPixelsPerInch

Allows overriding a pixel depth for drawing documents in this control (“pixel depth” is a count of logical pixels in one inch, also known as DPI or PPI)

```
property DocumentPixelsPerInch: Integer;
```

(introduced in version 18)

The actual pixel depth that used to render documents is returned by GetRealDocumentPixelsPerInch[319] method.

Zooming

DocumentPixelsPerInch allows to implement zooming. For example, if the screen (or the current monitor) DPI = 96, you can assign **DocumentPixelsPerInch** = 192 to apply 200% zoom.

This zooming affects only displaying in TRichView. It does not affect printing, exporting and importing to files.

After changing value of this property, call Format[295] or Reformat[339].

Zooming is applied to almost everything: margins, background images, items in a document. However, there are exceptions; the following items are not affected:

- controls[174]
- images from image lists (*bullets*[176] and *hot-spots*[178])*

You need to scale controls yourself, and assign different image lists.

* in Lazarus 2+, images from image lists are scaled as well (by choosing the proper image for the given DPI, or by scaling existing images of an image for this DPI is not included in TImageList).

RichViewActions note

If you use TRVRuler from RichViewActions, after changing a value of **DocumentPixelsPerInch**, you need to change the ruler's Zoom accordingly:

```
RVRuler1.Zoom :=
  Round(100 *
    RichViewEdit1.GetRealDocumentPixelsPerInch[319] /
    RVRuler1.ScreenRes);
```

If your application supports "per monitor v2", and if you changed **DocumentPixelsPerInch**, you need to recalculate the ruler's Zoom in the form's OnAfterMonitorDpiChanged event.

ScaleRichView note

There is an important difference between this type of zooming, and zooming in ScaleRichView.

In TRichView, when you change DocumentPixelsPerInch value, documents are rendered in different DPI, providing ideal results for this specific DPI. However, zooming is not exactly proportional (text width in DPI = 192 may not be exactly twice larger than text width in DPI = 96).

In ScaleRichView, documents are rendered in high DPI, and then are drawn on a screen or a printer, proportionally scaled. ScaleRichView may not provide ideal character positioning, but everything is scaled proportionally, allowing WYSIWYG implementation.

Default value:

0

See also:

- TRVStyle[739].UnitsPixelsPerInch[765]
- ZoomPercent[265] (FireMonkey analog)

6.1.1.2.22 TRichView.DoInPaletteMode

Defines behavior of TRichView when drawn in 256-color mode display.

```
type
  TRVPaletteAction =
    (rvpaDoNothing, rvpaAssignPalette,
     rvpaCreateCopies, rvpaCreateCopiesEx);

property DoInPaletteMode: TRVPaletteAction.
```

Note: TRVPaletteAction is declared in RVScroll unit.

This property affects only displaying in-256 color mode and does nothing in highcolor and truecolor modes.

This property may be considered obsolete since 256-color mode is rarely used.

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:

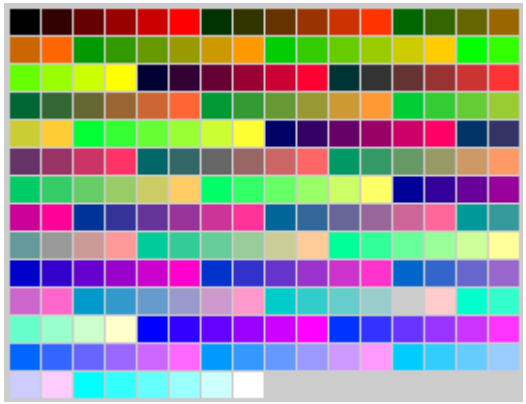
```
procedure TMyForm.WMDisplayChange(var Message: TWMDisplayChange);
begin
  // Call UpdatePaletteInfo methods of all your
  // RichViews, RichViewEdits, and RVPrints (if you use print preview)
  // for example:
  MyRichViewEdit.UpdatePaletteInfo(375);
  MyRVPrint.UpdatePaletteInfo(884);
end;
```

Default value:

rvpaCreateCopies

(or *rvpaDoNothing* in Delphi 2 and C++Builder 1)

TRichView 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(375).

See also methods of TRVPrint:

- UpdatePaletteInfo[884].

6.1.1.2.23 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[877] 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[244].

See also:

- Vertical scrolling[110].

6.1.1.2.24 TRichView.FirstJumpNo

Hypertext **id** for the first hyperlink in the document.

```
property FirstJumpNo: Integer;
```

When you append hypertext items (*hotspots*[178], *hot-pictures*[172] 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[390] event handler for several RichViews.

Default value:

0

See also:

- RichView hypertext[97].

6.1.1.2.25 TRichView.HTMLReadProperties

A set of properties controlling HTML import.

```
property HTMLReadProperties: TRVHTMLReaderProperties[432];
```

(introduced in version 21)

These properties are used by:

- TRichView.LoadHTML[330], LoadHTMLFromStream[332]
- TRichViewEdit.InsertHTMLFromFileEd[529], InsertHTMLFromStreamEd[530];
- TRichViewEdit.PasteHTML[545] (and Paste[543]);
- TRichView.LoadFromFile[328], LoadFromStream[329].

Read the topic about TRVHTMLReaderProperties[432] for details.

See also:

- HTMLSaveProperties [244]

6.1.1.2.26 TRichView.HTMLSaveProperties

A set of properties controlling HTML export.

```
property HTMLSaveProperties: TRVHTMLSaveProperties [436];
```

(introduced in version 21)

These properties are used by SaveHTML [342] and SaveHTMLToStream [346] methods.

Read the topic about TRVHTMLSaveProperties [436] for details.

See also:

- HTMLReadProperties [243]

6.1.1.2.27 TRichView.ItemCount

Number of items in RichView

```
property ItemCount: Integer;
```

See also:

- Item types [90];
- Adding items [107];
- Obtaining items [116];
- Modifying items [117].

6.1.1.2.28 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 [877] 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 [243].

See also:

- Vertical scrolling [110].

6.1.1.2.29 TRichView.LeftMargin

Left margin of document

```
property LeftMargin: TRVPixelLength[1085];
```

This value is measured in "standard" pixels, 1 "standard" pixel = 1/Style[260].UnitsPerInch[765] of an inch. When drawing or a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

Left margin can be used for selecting lines with the mouse.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

See also properties:

- RightMargin[251];
- TopMargin[262];
- BottomMargin[231].

See also indents in paragraph style[828].

See also properties of TRVPrint:

- Margins[705].

See also properties of TRVStyle:

- LineSelectCursor[753].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[110]

6.1.1.2.30 TRichView.LineCount

Obsolete, use ItemCount[244].

```
property LineCount : Integer;
```

See also

- GetLineNo[314].

6.1.1.2.31 TRichView.MarkdownProperties

A set of properties controlling ***Markdown*** export and import.

```
property MarkdownProperties: TRVMarkdownProperties[447];
```

(introduced in version 19)

Read the topic about TRVMarkdownProperties[447] for details.

Subproperties of this class affect the following methods:

- TRichView.SaveMarkdown[348], SaveMarkdownToStream[349], CopyMarkdown[289];
- TRichView.LoadMarkdown[332], LoadMarkdownFromStream[333];
- TRichView.LoadFromFile[328], LoadFromStream[329];
- TRichViewEdit.InsertMarkdownFromFileEd[532], InsertMarkdownFromStreamEd[532], PasteMarkdown[546].

6.1.1.2.32 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[265] = *False*. This property is ignored for paragraphs having *rvpaoNoWrap* in Options[833].

Default value:

0

6.1.1.2.33 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: TRVPixelLength[1085];
```

This property does not affect printing.

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style[260].UnitsPerInch[765] of an inch. When drawing or a screen, this value is recalculated according to the screen DPI (dots per inch) value.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Default value:

0

See horizontal scrolling[110] for details.

See also properties:

- MaxLength[246].

6.1.1.2.34 TRichView.MinTextWidth

Minimum width for text wrapping. This value allows to increase width of scrollable area in TRichView.

```
property MinTextWidth : TRVPixelLength[1085];
```

This property does not affect printing.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Default value:

0

See horizontal scrolling[110] for details.

6.1.1.2.35 TRichView.NoteText

Text for references to the parent note [190].

```
property NoteText: TRVUnicodeString[1099];
```

(introduced in version 10)

Normally, references [190] must be placed in Document [984] for footnotes [185], endnotes [185] or sidenotes [186]. In this case, they display the same text as their parent note.

References [190] 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 [984] for some note, you should set value of this property equal to NoteText [984] of this note.

Default value:

'?'

6.1.1.2.36 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,
    rvoDefaultSelectionFill*
  TRVOptions = set of TRVOption;

property Options: TRVOptions;
```

Note: TRVOption and TRVOptions are declared in RVScroll/fmxRVScroll unit.

Note*: this option is defined only for FireMonkey

Default value

For VCL and LCL Delphi/C++Builder 2009 or newer: [*rvoAllowSelection, rvoScrollToEnd, rvoAutoCopyUnicodeText, rvoAutoCopyImage, rvoAutoCopyRVF, rvoAutoCopyRTF, rvoDblClickSelectsWord, rvoRClickDeselects, rvoFormatInvalidate, rvoShowPageBreaks, rvoShowGridLines, rvoFastFormatting*]
Fore FireMonkey: the same + *rvoDefaultSelectionFill*.
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 [173] and tables [179]. See details [110]. Can be changed: any time, but you need to reformat [339] 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 [296] method is called. If cleared, document does not scroll in this case Can be changed: any time.
rvoSingleClick	If set, OnRVDblClick [399] 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 [295] and FormatTail [296] methods repaint component. Can be changed: any time.
rvoDisallowDrag	If set, dragging [123] 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 [418] 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 [905] • cell.BackgroundImageFileName [953]

Option	Meaning
	• *rvespImageFileName*[1072] item property See also: OnAssignImageFileName[379] 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[112].
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[469])

Show/Hide Options

Option	Meaning
rvoShowCheckpoints	If set, all *checkpoints* are shown (as dotted horizontal lines, or by OnDrawCheckpoint[776]). See details[92]. 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[168], soft hyphens in Unicode text, lines to left- and right-aligned items, places of insertion of text boxes[188]. Can be changed: any time, you need to reformat[339] RichView after changing this option, if the document is already formatted. See also: RVVisibleSpecialCharacters[1129], RichViewShowGhostSpaces[1116] typed constants.
rvoShowHiddenText	If set, hidden items[106] are displayed.

Option	Meaning
	Can be changed: any time, you need to reformat 339 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 1072 and OnItemHint 388 event.
rvoShowGridLines	If set, table 179 grid lines 750 are shown.

Clipboard Options (See Details 113)

[VCL, LCL, FMX for Windows and macOS]: If set , TRichView copies the selection to the Clipboard in the specified formats when the user presses Ctrl + Insert or Ctrl + C. These options also affect CopyDef 288. These options can be changed at any time.
[FMX for all platforms except for Windows and macOS]: the most appropriate format is chosen when copying.

Option	Format to Copy
rvoAutoCopyText	ANSI text (only for VCL and LCL)
rvoAutoCopyUnicodeText	Unicode text
rvoAutoCopyRVF	RVF 129 (RichView Format)
rvoAutoCopyRTF	RTF (Rich Text Format)
rvoAutoCopyImage	Bitmap or metafile (if the image is selected)

FireMonkey-specific Options

Option	Format to Copy
rvoDefaultSelectionFill	If set, and if 'selection' brush resource can be found in the linked visual style (specified in StyleLookup property), SelColor, SelOpacity, InactiveSelColor 751 properties of the linked 260 TRVStyle 739 component are updated according to this resource.

6.1.1.2.37 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[232][ItemNo] and ClearRight[232][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[247] (*rvoShowPageBreaks*).

See also methods:

- AssignSoftPageBreaks[284].

See also properties and events of TRVStyle:

- PageBreakColor[756];
- OnDrawPageBreak[778].

See also methods of TRichViewEdit:

- InsertPageBreak[534];
- RemoveCurrentPageBreak[551].

See also properties of table rows:

- PageBreakBefore[967].

6.1.1.2.38 TRichView.RightMargin

Right margin of document

```
property RightMargin: TRVPixelLength[1085];
```

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style[260].UnitsPerInch[765] of an inch. When drawing or a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

See also properties:

- LeftMargin[245];
- TopMargin[262];
- BottomMargin[231].

See also indents in paragraph style[828]

See also properties of TRVPrint:

- Margins[705].

See also properties of TRVStyle:

- LineSelectCursor[753].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[110]

6.1.1.2.39 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 [263]=*False*. If set (not recommended), RichView saves a style sheet in RTF; this style sheet is created from text and paragraph styles having Standard [804]=*True*. If UseStyleTemplates [263]=*True*, the component always saves Style [260].StyleTemplates [762] as a style sheet.
rvrtfDuplicateUnicode	If set, RichView saves an ANSI equivalent of Unicode text [135]. A conversion is based on Charset [809] 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 [1125].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.

Option	Meaning
	(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[1125].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[237] are saved in RTF and DocX.
rvrtfSaveHeaderFooter	If included, documents assigned by SetHeader[363] and .SetFooter[362] are saved as a header and footer in RTF and DocX.

Default value

[*rvrtfDuplicateUnicode*, *rvrtfSaveEMFAsWMF*, *rvrtfSaveJpegAsJpeg, rvrtfSavePngAsPng*]

See also methods:

- SaveRTF[350];
- SaveRTFToStream[351];
- CopyRTF[290].

See also:

- Saving and loading in TRichView[127];
- Pictures[169].

6.1.1.2.40 TRichView.RTFReadProperties

A set of properties controlling RTF (Rich Text Format) and DocX (Microsoft Word Document) import.

```
property RTFReadProperties: TRVRTFReaderProperties[457];
```

Read the topic about TRVRTFReaderProperties[457] for details.

Subproperties of this class affect the following methods:

- TRichView.LoadRTF[334], LoadDocX[326];
- TRichView.LoadRTFFromStream[335], LoadDocXFromStream[327];
- TRichView.LoadFromFile[328], LoadFromStream[329];
- TRichViewEdit.PasteRTF[547] (and Paste[543]);

- TRichViewEdit.InsertRTFFromStreamEd[536], InsertDocXFromStreamEd[527];
- TRichViewEdit.InsertRTFFromFileEd[537], InsertDocXFromFileEd[526];
- drag&drop[123] when dropping RTF file.

See also:

- RTFOptions[252] (options for RTF and DocX export).

6.1.1.2.41 TRichView.RVData

This read-only property points to an object containing RichView document.

Each TRichView control owns one such object.

Similar objects represent cells[951] in tables[901].

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[1014], 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[141].

6.1.1.2.42 TRichView.RVFOptions

Options for loading and saving RVF (RichView Format[129]). Some options also affect loading and saving XML files by TRichViewXML component.

```
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/fmxRVStyle unit.

Options for Saving

Option	Meaning
rvfoSavePicturesBody	▪ If set, full information about all pictures in RichView are saved in RVF. ▪ If not set, OnRVFPictureNeeded [401] event occurs on loading.
rvfoSaveControlsBody	▪ If set, full information about all controls in RichView are saved in RVF. ▪ If not set, OnRVFControlNeeded [400] 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 [764] property of the linked [260] TRVStyle component) is saved in RVF. See RVFTextStylesReadMode [258] property. This option affects saving style templates.*
rvfoSaveParaStyles	If set, collections of paragraph and list styles associated with this RichView (ParaStyles [757] and ListStyles [755] properties of the linked [260] TRVStyle component) are saved in RVF. See RVFParaStylesReadMode [258] property. This option affects saving style templates.*
rvfoSaveLayout	If set, the following properties are stored in RVF: Left [245]-, Right [251]-, Top [262]-, BottomMargin [231], MinTextWidth [246], MaxTextWidth [246], BiDiMode [230].

Option	Meaning
	Layout is saved only when saving the whole document (not selection). (see *rvfoLoadLayout* below)
rvfoSaveDocProperties	If set, DocProperties [238] and DocParameters [237] 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 [801] them itself elsewhere.
rvfoSaveDocObjects	If set, DocObjects [235] are saved in RVF. (see *rvfoLoadDocObjects* below)

* If the both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included, and UseStyleTemplates [263] =*True*, Style [260].StyleTemplates [762] are saved as well.

** This option also used by TRichViewXML component when saving XML files.

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 classes (unregistered classes and classes that cannot be detected by an image content). ▪ 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 [764].Count, or RichView.Style.ParaStyles [757].Count) style indices to 0.

Option	Meaning
	▪ 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 [238] and DocParameters [237] are loaded from RVF (but not by methods for RVF insertion) (see *rvfoSaveDocProperties* above)
rvfoCanChangeUnits	If set, the component can change Style [739].Units [764] to the value read from RVF. This change is possible only when RVF is loaded, not inserted.
rvfoLoadDocObjects	If set, DocObjects [235] 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 [1033]) 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 [169] and hot pictures [172]), background images of tables [904] and table cells [953]. For pictures of list markers [868], the specified graphic class is always used.

** This option also used by TRichViewXML component when loading XML files.

See RVF Overview [129] for details.

Default value (VCL and LCL):

[*rvfoSavePicturesBody, rvfoSaveControlsBody, rvfoSaveBinary, rvfoSaveDocProperties, rvfoLoadDocProperties, rvfoSaveDocObjects, rvfoLoadDocObjects*];

Default value (FireMonkey):

[*rvfoSavePicturesBody, rvfoSaveControlsBody, rvfoSaveBinary, rvfoSaveDocProperties, rvfoLoadDocProperties, rvfoSaveTextStyles, rvfoSaveParaStyles, rvfoSaveDocObjects, rvfoLoadDocObjects*];

Default value for components, placed on form at design time (can be changed in TRichView component editor) [224]:

[*rvfoSavePicturesBody, rvfoSaveControlsBody, rvfoSaveBinary, rvfoSaveDocProperties, rvfoLoadDocProperties, rvfoSaveTextStyles, rvfoSaveParaStyles, rvfoSaveDocObjects, rvfoLoadDocObjects*];

See also:

- RVFWarnings [258]

6.1.1.2.43 TRichView.RVFParaStylesReadMode

Specifies how paragraph styles [757] and list styles [755] are loaded from RVF files or streams.

```
property RVFParaStylesReadMode: TRVFReaderStyleMode[1076];
```

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode [1076] type for details.

Paragraph styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFTextStylesReadMode [258];
- RVFOptions [254].

See also:

- RVF Overview [129]

6.1.1.2.44 TRichView.RVFTextStylesReadMode

Specifies how text styles [764] are loaded from RVF files or streams.

```
property RVFTextStylesReadMode: TRVFReaderStyleMode[1076];
```

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode [1076] type for details.

Text styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFParaStylesReadMode [258];
- RVFOptions [254].

See also:

- RVF Overview [129].

6.1.1.2.45 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[254] 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[752].
rvfwUnknownCtrlProperties	There was error when reading inserted controls (for example, unknown properties)
rvfwConvUnits	RVF file has different units than Style[739].Units[764], and a conversion of units was made. This warning is not set if units were changed because of *rvfoCanChangeUnits* included in RVFOptions[254]. A conversion is performed automatically, without generating errors.

See RVF Overview[129] for details.

Methods reading RVF:

- LoadRVFFromStream[337];
- InsertRVFFromStream[323];
- AppendRVFFromStream[283];
- LoadRVF[336].

Methods of RichViewEdit reading RVF:

- InsertRVFFromFileEd[537];
- InsertRVFFromStreamEd[538];

(also PasteRVF[547] and Paste[543], but in this version you can't detect when Clipboard reading operation is finished).

6.1.1.2.46 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[403], if ssTouch is included in the Shift parameter of this event.

Default value:

True

See also properties of TRVStyle:

- SelectionHandleKind[757]

6.1.1.2.47 TRichView.SingleClick

```
property SingleClick : Boolean;
```

Obsolete, please use *rvoSingleClick* in Options[247];

6.1.1.2.48 TRichView.Style

Style of TRichView:

```
property Style: TRVStyle[739];
```

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.

6.1.1.2.49 TRichView.StyleTemplateInsertMode

Controls how Style[260].StyleTemplates[762] affect insertion of RTF, DocX, RVF, XML (by TRichViewXML component) files or streams.

```
type
  TRVStyleTemplateInsertMode =
    (rvstimUseTargetFormatting, rvstimUseSourceFormatting,
     rvstimIgnoreSourceStyleTemplates);
property StyleTemplateInsertMode: TRVStyleTemplateInsertMode;
```

(introduced in version 14)

This property is used only if UseStyleTemplates [263]=*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 [799] (if it exists). The inserted text is unstyled, except for hyperlinks [823] (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 [1111] = *False*. In the mode (3), "Normal" is always applied.

Default value

rvstimUseTargetFormatting

See also:

- Styles and style templates [103].

6.1.1.2.50 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[302] method.

6.1.1.2.51 TRichView.TopMargin

Top margin of document

```
property TopMargin: TRVPixelLength[1085];
```

This value is measured in "standard" pixels, 1 "standard" pixel = 1/Style[260].UnitsPerInch[765] of an inch. When drawing or a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

See also properties:

- LeftMargin[245];
- RightMargin[251];
- BottomMargin[231].

See also properties of TRVPrint:

- Margins[705].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[110]

6.1.1.2.52 TRichView.UseFMXThemes

Allows using a text color from a FireMonkey style.

```
property UseFMXThemes: Boolean;
```

(introduced in version 23)

If *True*, when drawing documents, FireMonkey style's foreground color is used instead of black color. All other colors are unaffected. FireMonkey style is specified in StyleLookup property.

If the specified style does not have TBrush resource named 'foreground', black color remains unchanged.

This property allows supporting light-on-dark styles.

Default value

False

See also:

- UseVCLThemes [264]
- DarkMode [234]

6.1.1.2.53 TRichView.UseStyleTemplates

Allows or disallows using Style [260].StyleTemplates [762].

```
property UseStyleTemplates: Boolean;
```

(introduced in version 14)

If UseStyleTemplates=*True*:

- StyleTemplates are saved in RVF (also requires both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* included in RVFOptions [254]);
- StyleTemplates are loaded from RVF (also requires RVFTextStylesReadMode [258] =RVFParaStylesReadMode [258]=*rvf_sInsertMerge*);
- StyleTemplates are saved in RTF and DocX (as a style sheet);
- StyleTemplates are loaded from RTF and DocX (also requires RTFReadProperties [253] .TextStyleMode [466]=RTFReadProperties.ParaStyleMode [464]=*rvrsAddIfNeeded*) ;
- StyleTemplates are saved in XML by TRichViewXML
- StyleTemplates are loaded from XML by TRichViewXML
- RTF, DocX, XML, and RVF files/streams are inserted according to StyleTemplateInsertMode [260];
- "Normal" style template (if it exists) is assigned to unstyled paragraphs when loading RTF and RVF files/streams (see RichViewAutoAssignNormalStyleTemplate [1111]);
- the component provides that TextStyle.ParaStyleTemplateId [815]=ParaStyle.StyleTemplateId [805] for all text items [167] and non-text items linked to a text style (tabs [168], labels [181], sequences [183], footnotes [185], endnotes [184], note references [190]); if they are not equal, the component assigns another text style to this item, with the correct ParaStyleTemplateId, adding this text style to Style [739].TextStyles [764] if necessary.
- in editor, when **Enter** is pressed, the next paragraph is formatted according to NextId [846] property of the current paragraph's style template

Default value

False

See also:

- Styles and style templates [103].

6.1.1.2.54 TRichView.UseVCLThemes

Allows using colors of VCL or custom themes.

```
property UseVCLThemes: Boolean;
```

(introduced in version 17)

If *True:*

- if RVGetSystemColor [1125] is defined, it is used to change colors in the document, otherwise
- all colors in document are changed according to the current VCL theme

Note: VCL themes are available in Delphi XE2 and newer.

Default value

False

See also:

- DarkMode [234]
- UseFMXThemes [263]

6.1.1.2.55 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 [286] method.

6.1.1.2.56 TRichView.VSmallStep

Number of pixels in one step of vertical scrollbar

```
property VSmallStep: Integer;
```

When VScrollPos [874] is changes by 1, the window is scrolled by VSmallStep pixels.

This is one "RVSU", see Scrolling in RichView [110]

Default value:

10

See also properties:

- VScrollPos [874];
- VScrollMax [874];
- VScrollVisible [874].

6.1.1.2.57 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 [833] of paragraph style, this property should be considered as a control (view) property, not as a document property. It is not stored in RVF [129].

Default value:

True

6.1.1.2.58 TRichView.ZoomPercent

Gets or sets the zoom percentage of the displayed document.

```
property ZoomPercent: Single;
```

(introduced in version 20)

This zooming affects only displaying in TRichView. It does not affect printing, exporting and importing to files.

After changing value of this property, call Format [295] or Reformat [339].

Zooming is applied to almost everything: margins, background images, items in a document. However, there are exceptions; the following items are not affected:

- controls [174].

You need to scale controls yourself.

Default value:

100 (meaning 100% zoom)

See also:

- DocumentPixelsPerInch [239] (analog for VCL)

6.1.1.3 Methods

Derived from TCustomRichView

- AddBreak [269]
- AddBullet [270]
- AddCheckpoint [271]
- AddControl [272]
- AddFmt [273]
- AddHotPicture [274]
- AddHotspot [275]
- AddItem [276]
- AddNL -A -W, Add [277]

AddPicture 278
AddTab 280
AddTextNL -A -W 281
AppendFrom 282
AppendRVFFromStream 283
AssignSoftPageBreaks 284
BeginOleDrag 284 [VCL, LCL]
Clear 286
ClearLiveSpellingResults 286
ClearSoftPageBreaks 286
ClientToDocument, DocumentToClient 286
ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits 287
Copy 288
CopyDef 288
CopyImage 289
CopyMarkdown 289
CopyRTF 290
CopyRVF 290
CopyText -A -W 290
Create 291
DeleteItems 291
DeleteMarkedStyles 292
DeleteParas 293
DeleteSection 293
DeleteUnusedStyles 293
Deselect 294
Destroy 294
FindCheckPointByName 294
FindCheckPointByTag 295
FindControlItemNo 295
Format 295
FormatTail 296
GetBreakInfo 296
GetBulletInfo 297
GetCheckpointByNo 298
GetCheckpointInfo 299
GetCheckpointItemNo 299
GetCheckpointNo 299
GetCheckpointXY 300
GetCheckpointY 300
GetCheckpointYEx 301
GetControlInfo 301
GetFirstCheckpoint 302
GetFocusedItem 302
GetHotspotInfo 303
GetItem 304
GetItemAt 305

GetItemCheckpoint 305
GetItemCoords 306
GetItemCoordsEx 307
GetItemExtraIntProperty 307
GetItemExtraIntPropertyEx 307
GetItemExtraStrProperty 308
GetItemExtraStrPropertyEx 308
GetItemNo 309
GetItemPara 309
GetItemStyle 310
GetItemTag 310
GetItemText -A -W 310
GetItemVAlign 311
GetJumpPointLocation 312
GetJumpPointY 313
GetLastCheckpoint 313
GetLineNo 314
GetListMarkerInfo 315
GetNextCheckpoint 315
GetOffsAfterItem 316
GetOffsBeforeItem 317
GetPictureInfo 318
GetPrevCheckpoint 319
GetRealDocumentPixelsPerInch 319
GetSelectedImage 319
GetSelectionBounds 320
GetSelText -W 321
GetTextInfo 321
GetWordAt -A -W 322
InsertRVFFromStream 323
IsFromNewLine 325
IsParaStart 325
LiveSpellingValidateWord 326
LoadDocX 326 (D2009+ or with Delphi ZLib)
LoadDocXFromStream 327 (D2009+ or with Delphi ZLib)
LoadFromFile 328
LoadFromFileEx 328
LoadFromStream 329
LoadFromStreamEx 329
LoadHTML 330
LoadHTMLFromStream 332
LoadMarkdown 332
LoadMarkdownFromStream 333
LoadRTF 334
LoadRTFFromStream 335
LoadRVF 336
LoadRVFFromStream 337

LoadText -W 338
LoadTextFromStream -W 339
MarkStylesInUse 339
RefreshListMarkers 340
Reformat 339
RemoveCheckpoint 340
ResetAnimation 341
SaveDocX 341
SaveDocXToStream 342
SaveHTML 342
SaveHTMLToStream 346
SavePicture 349
SaveMarkdown 348
SaveMarkdownToStream 349
SaveRTF 350
SaveRTFToStream 351
SaveRVF 351
SaveRVFToStream 352
SaveText -W 352
SaveTextToStream -W 353
SearchText -A -W 354
SelectAll 355
SelectControl 355
SelectionExists 356
SelectWordAt 356
SetAddParagraphMode 357
SetBreakInfo 358
SetBulletInfo 359
SetCheckpointInfo 360
SetControlInfo 361
SetFooter 362
SetHeader 363
SetHotspotInfo 364
SetItemExtraIntProperty 366
SetItemExtraIntPropertyEx 366
SetItemExtraStrProperty 367
SetItemExtraStrPropertyEx 367
SetItemTag 367
SetItemText -A -W 368
SetItemVAlign 369
SetListMarkerInfo 369
SetPictureInfo 370
SetSelectionBounds 372
StartAnimation 374
StartLiveSpelling 374
StopAnimation 375
StoreSearchResult 375

UpdatePaletteInfo [375]

Derived from TRVScroller [869]

ScrollTo [875]

6.1.1.3.1 TRichView.AddBreak

Adds a *break* [173] (a horizontal line) to the end of document.

```
procedure AddBreak(Width: TRVStyleLength[1094] = 0;
  Style: TRVBreakStyle[1059] = rvbsLine; Color: TRVColor[1063] = rvclNone[1105];
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
```

(changed in version 18)

Parameters:

Width – line width (or rectangle height). This value is measured in Style [260].Units [764]. If **Width** = 0, the new item has 1 pixel width (converted to Style [260].Units [764])

Style – break type;

Color – color of line. If it is equal to *rvclNone* [1105], the *break* color is Style [260].TextStyles [764] [0].Color [810].

Tag – *tag* [96] of this *break*.

For default HTML <hr> appearance, parameters must be: **Style**=*rvbs3d*, **Width**=2.

Methods type: viewer-style.

See also methods:

- Format [295];
- FormatTail [296].

See also methods of TRichViewEdit:

- InsertBreak [523].

See also:

- Item types [90];
- Other methods for appending items [107];
- "Tags" [96].

Related deprecated methods:

```
procedure AddBreakExTag(Width: TRVStyleLength[1094]; Style: TRVBreakStyle[1059];
  Color: TRVColor[1063]; const Tag: TRVTag[1096]);
procedure AddBreakEx(Width: TRVStyleLength[1094]; Style: TRVBreakStyle[1059];
  Color: TRVColor[1063]);
procedure AddBreakTag(const Tag: TRVTag[1096]);
```

These methods provide subset of functionality of AddBreak.

6.1.1.3.2 TRichView.AddBullet

Adds a *bullet*[176] (an image from TImageList) to the end of document.

VCL and LCL:

```
procedure AddBullet(const AName: TRVUnicodeString[1099];
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
```

(changed in version 18)

FireMonkey:

```
procedure AddBullet(const AName: TRVUnicodeString[1099];
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124];
  ImageWidth: TRVStyleLength[1094] = 16;
  ImageHeight: TRVStyleLength[1094] = 16);
```

Alternatives:

- If you want to add a hyperlink-picture from image list, add a *hotspot*[275] instead.
- If you want to add a normal picture, add a picture[278] instead.
- If you want to add a normal picture-hyperlink, add a *hot picture*[274] instead.

Parameters:

AName – name of *bullet*, any string. **AName** 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[260].ParaStyles[757][**ParaNo**].

Tag – *tag*[96] of this *bullet*.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Method type: viewer-style.

See also methods:

- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also methods of TRichViewEdit:

- InsertBullet[524].

See also properties of TRVStyle:

- ParaStyles[757].

See also:

- Item types [90];
- Other methods for appending items [107];
- "Tags" [96].

Related deprecated methods:

```
procedure AddBulletExTag(const Name: TRVUnicodeString[1099];
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer; const Tag: TRVTag[1096]);
procedure AddBulletEx(const Name: TRVUnicodeString[1099];
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer);
```

These methods provide subset of functionality of AddBullet.

6.1.1.3.3 TRichView.AddCheckpoint

Adds a *checkpoint* [92] to the end of document.

```
function AddCheckpoint(const CpName: TRVUnicodeString[1099] = '';
  RaiseEvent: Boolean = False;
  const CpTag: TRVTag[1096] = RVEMPTYTAG[1124]): 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 [379]

Tag – *tag* of *checkpoint*.

Return value:

Index of the added checkpoint

Methods type: viewer-style.

See also methods:

- SetCheckpointInfo [360];
- RemoveCheckpoint [340].

See also methods of TRichViewEdit:

- SetCheckpointInfoEd [560];
- SetCurrentCheckpointInfo [563].

See also:

- Methods for appending items [107];
- *Checkpoints* [92];
- *Tags* [96].

Related deprecated methods:

```
function AddNamedCheckpointExTag(const CpName: TRVUnicodeString[1099];
```

```
  RaiseEvent: Boolean; const Tag: TRVTag[1096]): Integer;
function AddCheckpoint: Integer;
function AddCheckpointTag(const Tag: TRVTag[1096]): Integer;
function AddNamedCheckpoint(const CpName: TRVUnicodeString[1099]): Integer;
function AddNamedCheckpointEx(const CpName: TRVUnicodeString[1099];
  RaiseEvent: Boolean): Integer;
```

These methods provide subset of functionality of AddCheckpoint.

6.1.1.3.4 TRichView.AddControl

These methods add Delphi control[174] to the end of document.

```
procedure AddControl(const Name: TRVUnicodeString[1099]; ctrl: TControl;
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
```

(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[1100] 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[260].ParaStyles[757][**ParaNo**].

Tag – *tag*[96] of this control item. Do not confuse with **ctrl**.Tag property.

Methods type: viewer-style.

See also methods:

- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also methods of TRichViewEdit:

- InsertControl[526].

See also properties of TRVStyle:

- ParaStyles[757].

See also:

- Item types[90];
- Other methods for appending items[107];
- "Tags"[96].

Related deprecated methods:

```
procedure AddControlExTag(const Name: TRVAnsiString[1056]; ctrl: TControl;
  ParaNo: Integer; VAlign: TRVVAlign[1100]; const Tag: TRVTag[1096]);
procedure AddControlEx(const Name: TRVAnsiString[1056]; ctrl: TControl;
```

```
  ParaNo: Integer; VAlign: TRVVAlign[1100]);
```

These methods provide subset of functionality of AddControl.

6.1.1.3.5 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[277](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[143].

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[281].

Args – arguments for format string.

StyleNo is an index in the TextStyles[764] collection of the linked[260] RVStyle component, or rvsDefStyle[1127] 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[260].ParaStyles[757][**ParaNo**].

Method type: viewer-style.

Unicode note: the parameter of this method is:

- Unicode string (UTF-16) for Delphi/C++Builder 2009 or newer,
- ANSI string for older versions of Delphi/C++Builder.
- Unicode string (UTF-8) in Lazarus[160]

Example:

```
MyRichView.AddFmt('X=%d and Y=%d', [X,Y], 0, 0);
```

If style templates[263] 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[277];
- AddTextNL[281];
- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also properties of TRVStyle:

- TextStyles[764];
- ParaStyles[757];

- SpacesInTab[761].

See also:

- Other methods for appending items[107].

6.1.1.3.6 TRichView.AddHotPicture

Adds picture-hyperlink[172] to the end of document.

```
procedure AddHotPicture(const Name: TRVUnicodeString[1099];
  gr: TRVGraphic[1032]; ParaNo: Integer = -1;
  VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124];
  ImageWidth: TRVStyleLength[1094] = 0;
  ImageHeight: TRVStyleLength[1094] = 0);
```

(introduced in version 1.5; changed in versions 18 and 21)

The same as AddPicture[278], 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[1067], 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[1114]). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign[1100] 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[260].ParaStyles[757][**ParaNo**].

Tag – *tag*[96] of this *hot-picture* item..

The image is displayed stretched to **ImageWidth** x **ImageHeight** (if they are non-zero, see *rvepImageWidth* and *rvepImageHeight* in IDH_Type_TRVExtraItemProperty[1067]).

Alternatives:

- If you want to add a simple (not a hyperlink) picture, add a picture[278] instead.
- If you want to add a hyperlink-picture from image list, add a *hotspot*[275] instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet*[270] instead.

Methods type: viewer-style.

See also methods:

- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also methods of TRichViewEdit:

- InsertHotPicture[528].

See also properties of TRVStyle:

- ParaStyles [757].

See also:

- Item types [90];
- Other methods for appending items [107];
- "Tags" [96];
- Hypertext in TRichView [97].

Related deprecated methods:

```
procedure AddHotPictureTag(const Name: TRVUnicodeString[1099];
  gr: TGraphic; ParaNo: Integer;
  VAlign: TRVVAlign[1100]; const Tag: TRVTag[1096]);.
```

6.1.1.3.7 TRichView.AddHotspot

Adds a *hotspot* [178] (an image from TImageList – hypertext link) to the end of document.

VCL and LCL:

```
procedure AddHotspot(const AName: TRVUnicodeString[1099];
  ImageIndex, HotImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
```

(changed in version 18)

FireMonkey:

```
procedure AddHotspot(const AName: TRVUnicodeString[1099];
  ImageIndex, HotImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124];
  ImageWidth: TRVStyleLength[1094] = 16;
  ImageHeight: TRVStyleLength[1094] = 16);
```

Alternatives:

- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet* [270] instead.
- If you want to add a picture-hyperlink, add a *hot picture* [274] instead.
- If you want to add a picture, add a picture [278] instead.

Parameters:

AName – name of *hotspot*, any string. **AName** 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[260].ParaStyles[757][**ParaNo**].

Tag – *tag*[96] of this *hotspot*.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Methods type: viewer-style.

See also methods:

- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also methods of TRichViewEdit:

- InsertHotspot[528].

See also properties of TRVStyle:

- ParaStyles[757].

See also:

- Item types[90];
- Other methods for appending items[107];
- "Tags"[96];
- Hypertext in RichView[97].

Related deprecated methods:

```
procedure AddHotspotExTag(const Name: TRVUnicodeString[1099];
  ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer;
  const Tag: TRVTag[1096]);
procedure AddHotspotEx(const Name: TRVUnicodeString[1099];
  ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer);
```

These methods provide subset of functionality of AddHotspot.

6.1.1.3.8 TRichView.AddItem

Adds one item to the end of document.

```
procedure AddItem(const Text: TRVUnicodeString[1099]; Item: TCustomRVItemInfo[899]);
```

(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[107].

Usually this method is used for appending tables[179], labels[181], sequences[183], footnotes[185], endnotes[184], references to notes[190].

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 [899].

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 [757])

Method type: viewer-style.

See also methods:

- Format [295];
- FormatTail [296];
- SetAddParagraphMode [357];
- GetItem [304].

See also methods of TRichViewEdit:

- InsertItem [531].

See also:

- Item types [90];
- Other methods for appending items [107].

6.1.1.3.9 TRichView.AddNL -A -W

These methods add one text item [167] 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[1096] = RVEMPTYTAG[1124]);
procedure AddNLW(const s: TRVUnicodeString[1099]; StyleNo, ParaNo: Integer;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
procedure AddNLA(const s: TRVAnsiString[1056]; StyleNo, ParaNo: Integer;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]);
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 [281].

StyleNo is an index in the TextStyles [764] collection of the linked [260] RVStyle component, or *rvsDefStyle* [1127] 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 [260].ParaStyles [757][**ParaNo**].

Tag – *tag* [96] 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 [143].

Unicode notes:

Internally, text is stored as Unicode. **AddNLA** converts ANSI to Unicode. A code page for conversion is calculated basing on Style [260].TextStyles [764][**StyleNo**]. Charset [809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style [260].DefCodePage [745] is used.

AddNL works:

- like **AddNLA**, in Delphi 2007 and older
- like **AddNLW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus [160]

Methods type: viewer-style

If style templates [263] 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 [280];
- Format [295];
- FormatTail [296];
- SetAddParagraphMode [357].

See also methods of TRichViewEdit:

- InsertStringTag, InsertStringATag, InsertStringWTag [539].

See also properties of TRVStyle:

- TextStyles [764];
- SpacesInTab [761].

See also:

- Other methods for appending items [107];
- Unicode in RichView [135].

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)
```

6.1.1.3.10 TRichView.AddPicture

Adds a picture [169] to the end of document

```
procedure AddPicture(const Name: TRVUnicodeString[1099]; gr: TRVGraphic[696];
  ParaNo: Integer = -1; VAlign: TRVVAlign[1100] = rvvaBaseline;
  const Tag: TRVTag[1096] = RVEMPTYTAG[1124]; ImageWidth: TRVStyleLength[1094] = 0;
 ImageHeight: TRVStyleLength[1094] = 0);
```

(changed in versions 18 and 21)

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[1067], 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[1114]). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign[1100] 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[260].ParaStyles[757][**ParaNo**].

Tag – *tag*[96] of this picture item.

The image is displayed stretched to **ImageWidth** x **ImageHeight** (if they are non-zero, see *rvepImageWidth* and *rvepImageHeight* in IDH_Type_TRVExtraItemProperty[1067]).

Alternatives:

- If you want to add a picture-hyperlink, add a *hot picture*[274] instead.
- If you want to add a hyperlink-picture from image list, add a *hotspot*[275] instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet*[270] 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[286];
AddBitmapFromFile(rv, 'd:\pic1.bmp', 0);
AddBitmapFromFile(rv, 'd:\pic2.bmp', -1);
rv.Format[295];
```

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[366](rv.ItemCount[244]-1, rvepShared, 1);
```

```
rv.AddPicture('', bmp, -1);
rv.SetItemExtraIntProperty(366)(rv.ItemCount(244)-1, rvepShared, 1);
rv.Format(295);
// you must free bmp yourself when rv is cleared.
```

See also methods:

- Format(295);
- FormatTail(296);
- SetAddParagraphMode(357).

See also methods of TRichViewEdit:

- InsertPicture(534).

See also properties of TRVStyle:

- ParaStyles(757).

See also:

- Item types(90);
- Other methods for appending items(107);
- "Tags"(96).

Related deprecated methods:

```
procedure AddPictureExTag(const Name: TRVUnicodeString(1099); gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign(1100); const Tag: TRVTag(1096));
procedure AddPictureEx(const Name: TRVUnicodeString(1099); gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign(1100));
```

These methods provide subset of functionality of AddPicture.

6.1.1.3.11 TRichView.AddTab

Adds tabulator(168) 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(764) collection of the linked(260) RVStyle component, or *rvsDefStyle*(1127) 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(260).ParaStyles(757)[**ParaNo**].

This method always adds a tabulator, regardless of RVStyle.SpacesInTab(761) mode. If RVStyle.SpacesInTab(761)=0, this method does the same work as AddTextNL(281)(#9, TextStyleNo, ParaNo, ParaNo).

Methods type: viewer-style.

See also methods:

- Format(295);
- FormatTail(296);
- SetAddParagraphMode(357).

See also methods of TRichViewEdit:

- InsertTab [540].

See also properties of TRVStyle:

- ParaStyles [757].

See also:

- Item types [90];
- Other methods for appending items [107].

6.1.1.3.12 TRichView.AddTextNL -A -W

These methods add one or more text [167] lines to the end of document.

```
procedure AddTextNL(const s: String;
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag [1096] = RVEMPTYTAG [1124]);
procedure AddTextNLA(const s: TRVAnsiString [1056];
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag [1096] = RVEMPTYTAG [1124];
  CodePage: TRVCodePage = CP_ACP);
procedure AddTextNLW(const S: TRVUnicodeString [1099];
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  DefAsSingleParagraph: Boolean = False;
  const Tag: TRVTag [1096] = RVEMPTYTAG [1124]);
```

(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 [168], depending on value of SpacesInTab [761] property of the linked [260] RVStyle component. #12 characters add page breaks [250]. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported.

StyleNo is an index in the TextStyles [764] collection of the linked [260] RVStyle component, or *rvsDefStyle* [1127] 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 [260].ParaStyles [757][**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 [357]). 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 [357]). If *True*, it treats #13 and #10 as line breaks, not paragraph breaks.

Tag – *tag* [96] 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** converts ANSI string to Unicode using **CodePage**.

If CodePage = CP_ACP, Style[260].DefCodePage[745] is used instead.

AddTextNL works:

- like **AddTextNLA**, in Delphi 2007 and older
- like **AddTextNLW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus[160]

If style templates[263] 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[277];
- AddTab[280];
- LoadText, LoadTextW[338];
- LoadTextFromStream, LoadTextFromStreamW[339];
- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357].

See also methods of TRichViewEdit:

- InsertText, -A, -W[540].

See also properties of TRVStyle:

- TextStyles[764];
- SpacesInTab[761].

See also:

- Other methods for appending items[107];
- Unicode in RichView[135];
- Example how to load UTF-8 files[468].

6.1.1.3.13 TRichView.AppendFrom

Adds content of another RichView component (**Source**) to the end of document.

```
procedure AppendFrom(Source: TCustomRichView[216]);
```

This method does not copy items of some types (including inserted controls[174] and tables[179]) (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(Stream, False);
    Stream.Position := 0;
    Dest.LoadRVFFromStream(Stream);
    Dest.Format;
  finally
    Stream.Free;
  end;
end;
```

Method type: viewer-style.

See also methods:

- Format;
- FormatTail.

See also:

- Other methods for appending items.

6.1.1.3.14 TRichView.AppendRVFFromStream

Adds data from Stream in RichView Format (RVF) 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 collection of the linked 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.

If style templates are used, and RVFTextStylesReadMode=RVFParaStylesReadMode=*rvf_sInsertMerge*, and StyleTemplateInsertMode<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style.StyleTemplates, and reads text and paragraph styles according to StyleTemplateInsertMode. The method calls OnStyleTemplatesChange event.

Return value:

"Was reading successful?"

See also properties:

- RVFOptions;
- RVFWarnings;
- RVFTextStylesReadMode (this method type: RVF insertion);

- RVFParaStylesReadMode[258] (this method type: RVF insertion);
- UseStyleTemplates[263].

See also events:

- OnRVFImageListNeeded[400];
- OnRVFControlNeeded[400];
- OnRVFPictureNeeded[401];
- OnControlAction[380] (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange[417].

See also methods:

- InsertRVFFromStream[323];
- LoadRVFFromStream[337];
- LoadRVF[336];
- Format[295];
- FormatTail[296].

See also:

- TRichView component editor[224];
- Saving and loading[127];
- RVF overview[129];
- Paragraphs[100].

6.1.1.3.15 TRichView.AssignSoftPageBreaks

Displays soft page breaks (i.e. page breaks that were made automatically)

```
procedure AssignSoftPageBreaks(RVPrint[696]: TComponent);
```

(introduced in version 1.7)

Page breaks become visible after calling RichView.AssignSoftPageBreaks(RVPrint). **RVPrint** must be formatted (i.e. FormatPages[711] must be called before this method). Page breaks are visible only if *rvoShowPageBreaks* is included in RichView.Options[247].

In editor[469], soft page breaks will be automatically cleared after any editing operation (any action invoking OnChange[582]). Page breaks will also be cleared when you call Clear[286]. In all other cases, you must call ClearSoftPageBreaks[286] after each change in the document performed after calling AssignSoftPageBreaks.

Soft page breaks are drawn using RVStyle.SoftPageBreakColor[761] (or in OnDrawPageBreak[778] event).

See also:

- ClearSoftPageBreaks[286].

See also properties:

- PageBreaksBeforeItems[250].

6.1.1.3.16 TRichView.BeginOleDrag

Starts dragging the selection.

```
procedure BeginOleDrag;
```

(introduced in v1.9)

This method is useful if you want to implement drag&drop [123] of inserted controls [174].(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 [355](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 [526]('',btn,rvvaBaseline);
```

4) Events are not saved in RVF, so reassign them on loading. Use OnControlAction [380] 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:
https://www.trichview.com/forums/viewtopic.php?t=157

6.1.1.3.17 TRichView.Clear

Clears the document.

```
procedure Clear; virtual;
```

This method:

- deletes all items [90];
- turns off displaying soft page breaks [286];
- clears DocProperties [238];
- clears DocObjects [235];
- stops live spelling check [286];
- stops animations [124], if AnimationMode [228] = *rvaniOnFormat*.

This method does not change collections of text, paragraph and list styles. So, if your application adds styles while editing, call DeleteUnusedStyles [293] 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 [336], LoadRVFFromStream [337].

See also methods:

- Format [295].

See also method of TRichViewEdit:

- Clear [509].

See also:

- Methods for building documents [107].

6.1.1.3.18 TRichView.ClearLiveSpellingResults

Clears all live spelling underlines and stops the live spelling thread.

```
procedure ClearLiveSpellingResults;
```

(introduced in v1.9)

Clear [286] method calls this procedure automatically.

See also: Live spelling check in RichView [138].

6.1.1.3.19 TRichView.ClearSoftPageBreaks

Turns off displaying soft page breaks

```
procedure ClearSoftPageBreaks;
```

See also:

- AssignSoftPageBreaks [284].

6.1.1.3.20 TRichView.ClientToDocument, DocumentToClient

The methods convert client coordinates to document coordinates and vice versa.

```
function ClientToDocument(const APoint: TRVCoordPoint[1064]): TRVCoordPoint[1064];
```

(introduced in v1.9)

```
function DocumentToClient(const APoint: TRVCoordPoint[1064]): TRVCoordPoint[1064];
```

(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[305] uses this coordinate system.

The methods take a cell rotation[961] into account.

Note:

This method must be used instead of adding HScrollPos[872] to X and VScrollPos[874]*VSmallStep[264] to Y. The old method was used in some old example code, it is correct only if the document is top-aligned[264].

See also:

Example in OnRVMouseDown[402].

6.1.1.3.21 TRichView.ConvertDocTo...

The methods convert all lengths in the current document to different units of measurement.

```
procedure ConvertDocToPixels;
procedure ConvertDocToTwips;
procedure ConvertDocToEMU;
procedure ConvertDocToDifferentUnits(NewUnits: TRVStyleUnits[1094]);
```

(introduced in version 13)

The methods convert all lengths in the current document from Style[260].Units[764] to different units of measurement:

- pixels (1 pixel = 1/Style[260].UnitsPixelsPerInch[765] of an inch)
- twips,
- EMU
- NewUnits (pixels, twips, or EMU),

respectively.

The methods convert only properties of items[163], they do not convert properties of the Style[260] itself.

To convert the document completely:

- call one of these methods (for all documents linked to the same TRVStyle[739] component);
- call one of Style[260].ConvertTo*[767] method to convert properties of Style[260] to the same units of measurement.

See also:

- Units of measurement in TRichView[144]

6.1.1.3.22 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.

Copying [VCL; Lazarus; FMX for Windows, macOS, and Linux]

The selected fragment is copied to the Clipboard in all formats.

Copying [FMX for Android and iOS]

Calls CopyDef [288] to copy in the most appropriate format.

See also:

- Working with selection [112];
- Working with Clipboard [113].

6.1.1.3.23 TRichView.CopyDef

Copies the selected fragment to the Clipboard in several formats according to Options [247].

```
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.

On Windows, 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.

This method must be called only when the document is formatted.

Copying [VCL; Lazarus; FMX for Windows, macOS, and Linux]

The selected fragment is copied to the Clipboard in multiple formats.

- if (*rvoAutoCopyRVF* in Options), it copies selection as RVF [129].
- 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 [135] text.
- if (*rvoAutoCopyRTF* in Options), it copies selection as RTF.

Copying [FMX for Android and iOS]

The selected fragment is copied to the Clipboard in the most appropriate format.

- if (*rvoAutoCopyImage* in Options), and there is only one picture selected, it copies selection as an image; otherwise
- if (*rvoAutoCopyRVF* in Options), it copies selection as RVF[129]; otherwise
- if (*rvoAutoCopyRTF* in Options), it copies selection as RTF; otherwise
- if (*rvoAutoCopyUnicodeText* in Options), it copies text as Unicode[135] text.

Return value:

"Is the selection not empty?" and "Is at least one of *rvoAutoCopy**** options set?"

See also:

- Working with selection[112];
- Working with Clipboard[113].

6.1.1.3.24 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[112];
- Working with Clipboard[113].

6.1.1.3.25 TRichView.CopyMarkdown

Copies the selected fragment to the Clipboard as text in Markdown format.

```
procedure CopyMarkdown;
```

(Introduced in version 24)

The Clipboard does not have a special format identifier for Markdown, so this method copies Markdown as a plain text. This is an alternative to CopyText[290].

TRichView does not copy Markdown by default (see CopyDef[288]). If you want to use this method, implement “Copy as Markdown” command and call this method from it.

Parameters for writing Markdown are in TRichView.MarkdownProperties[245].

This method must be called only when the document is formatted.

See also:

- Working with selection[112];

- Working with Clipboard [113].

6.1.1.3.26 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 [224].

Options for saving are in RTFOptions [252] property.

By default, items tags [96] are saved as hyperlinks targets. You can customize saving using OnWriteHyperlink [418] event.

You can save additional information about non-text objects in OnWriteObjectProperties [420] event.

This method must be called only when the document is formatted.

See also:

- SaveRTF [350], SaveRTFToStream [351] methods;
- RTFOptions [252] property;
- UseStyleTemplates [263] property;
- Working with selection [112];
- Working with Clipboard [113].

6.1.1.3.27 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 [224].

This method must be called only when the document is formatted.

See also:

- Working with selection [112];
- Working with Clipboard [113];
- RVF [129].

6.1.1.3.28 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.

CopyTextW and **CopyText** copy text in Unicode encoding.

CopyTextA copies text in ANSI encoding. Text of Unicode styles is converted basing on TRVStyle.DefCodePage[745].

Since modern versions of Windows converts Unicode and ANSI texts in the Clipboard to each other, using **CopyTextA** does not make sense (and it is not available in non-Windows versions).

These methods must be called only when the document is formatted.

Alternatively, you can copy text as Markdown using CopyMarkdown[289] method.

See also:

- Working with selection[112];
- Working with Clipboard[113];
- Unicode[135].

6.1.1.3.29 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[224] (and overridden by default!)

6.1.1.3.30 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[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].DeleteItems.

Method type: viewer-style.

Be careful: this method can produce invalid document[143].

Example 1: deleting all empty lines

```
procedure DeleteBlankLines(RVData: TCustomRVData[1014]);
var i,r,c: Integer;
    table: TRVTableItemInfo[901];
begin
  for i := RVData.ItemCount[244]-1 downto 0 do
    if RVData.IsParaStart[325](i) and (RVData.GetItemStyle[310](i)>=0) and
      (RVData.GetItemTextR(i)='') and (RVData.ItemCount[244]>1) then
      RVData.DeleteItems(i,1)
    else if RVData.GetItem[304](i) is TRVTableItemInfo[901] then
```

```
    begin
      table := TRVTableItemInfo(RVData.GetItem(i));
      for r := 0 to table.RowCount-1 do
        for c := 0 to table.ColCount-1 do
          if table.Cells[r,c]<>nil then
            DeleteBlankLines(table.Cells[r,c].GetRVData);
    end;
end;
```

DeleteBlankLines cannot be undone/redone. If called for editor, the editor's undo buffer becomes incorrect, so call ClearUndo.

Call:

```
DeleteBlankLines(RichViewEdit1.RVData);
RichViewEdit1.ClearUndo;
RichViewEdit1.Format;
```

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;
- DeleteParas.

6.1.1.3.31 TRichView.DeleteMarkedStyles

Deletes unused text, paragraph and list styles.

```
procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData);
```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents (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 method, see TRVDeleteUnusedStylesData 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 instead.

See also methods:

- MarkStylesInUse;
- DeleteUnusedStyles.

6.1.1.3.32 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 [296], it allows updating chat windows faster, without complete reformatting.

Items are indexed from 0 to ItemCount [244]-1..

Method type: unique method. Unlike all other methods of TRichView, it quickly reformats TRichView, so you do not need to call Format [295] 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 [291];
- DeleteSection [293].

6.1.1.3.33 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* [92] 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 [143].

See also:

- DeleteItems [291];
- DeleteParas [293].

6.1.1.3.34 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 [764], ParaStyles [757], ListStyles [755]) of the linked [260] 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[804] property of styles)

This method does not require reformatting.

See also methods:

- MarkStylesInUse[339];
- DeleteMarkedStyles[292].

6.1.1.3.35 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[416] event.

See also methods:

- SetSelectionBounds[372];
- SelectAll[355].

See also:

- Selecting in RichView document[112].

6.1.1.3.36 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.

6.1.1.3.37 TRichView.FindCheckpointByName

Returns the first *checkpoint* having the specified **Name**, or **nil** if there is no such *checkpoint*

```
function FindCheckpointByName(
  const Name: TRVUnicodeString[1099]): TCheckpointData[1055];
```

See also methods:

- FindCheckpointByTag [295];

See also:

- "Checkpoints" [92].

6.1.1.3.38 TRichView.FindCheckpointByTag

Returns the first checkpoint having the specified tag, or nil if there is no such checkpoint

```
function FindCheckpointByTag(const Tag: TRVTag [1096]): TCheckpointData [1055];
```

Example:

```
var
  CheckpointData: TCheckpointData;
...
  CheckpointData := MyRichView.FindCheckpointByTag(
    'Tag_String_Example');
```

See also methods:

- FindCheckpointByName [294].

See also:

- "Tags" [96];
- "Checkpoints" [92].

6.1.1.3.39 TRichView.FindControlItemNo

Returns an index of the RichView item which owns the control [174] **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 [90].

6.1.1.3.40 TRichView.Format

Formats the document. Only formatted documents can be displayed or edited.

```
procedure Format;
```

If *rvoFormatInvalidate* in Options [247] (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 [469] 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[296];
- Reformat[339].

See also properties:

- Options[247].

See also:

- Building document[107].

6.1.1.3.41 TRichView.FormatTail

Formats a part of document appended after the last call of Format[295] 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[295] 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[293], it allows updating chat windows faster, without complete reformatting.

If *rvoFormatInvalidate* in Options[247], this method also repaints RichView.

See also methods:

- Format[295];
- DeleteParas[293].

See also properties:

- Options[247];
- VScrollVisible[874].

See also:

- Appending items[107].

6.1.1.3.42 TRichView.GetBreakInfo

Returns main properties of item of *break* type[173]

```
procedure GetBreakInfo(ItemNo: Integer; out AWidth: TRVStyleLength[1094];
  out AStyle: TRVBreakStyle[1059]; out AColor: TRVColor[1063];
  out ATag: TRVTag[1096]);
```

Input parameter:

ItemNo – index of the item. The item must be of *break*[173] type (*rvsBreak*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetBreakInfo.

Output parameters:

AWidth – line width (or rectangle height). This value is measured in Style[260].Units[764].

AStyle – visual style of this *break*, see TRVBreakStyle[1059] for possible values.

AColor – line color. If it is equal to *rvclNone*[1105], Style[260].TextStyles[764][0].Color[810] is used.

ATag – *tag* of the item. Use SetItemTag[367] or SetBreakInfo[358] to change *tag* of item.

Instead of this method, you can use GetItemExtraIntPropertyEx[307] and GetItemTag[310] methods.

Additional item properties are returned by the methods GetItemExtraIntProperty[307] and GetItemExtraStrProperty[308].

See also methods:

- SetBreakInfo[358];
- GetItemStyle[310];
- GetItemExtraIntProperty[307];
- GetItemExtraStrProperty[308].

See also properties:

- ItemCount[244].

See also:

- Obtaining RichView items[116];
- Item types[90];
- *Tags*[96].

6.1.1.3.43 TRichView.GetBulletInfo

Returns main properties of item of *bullet* type[176]

VCL and LCL:

```
procedure GetBulletInfo(ItemNo: Integer; out AName: TRVUnicodeString[1099];
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure GetBulletInfo(ItemNo: Integer; out AName: TRVUnicodeString[1099];
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag[1096];
  out ImageWidth, ImageHeight: TRVStyleLength[1094]);
```

Input parameter:

ItemNo – index of the item. The item must be of *bullet*[176] or *hotspot*[178] type (*rvsBullet*[1127] or *rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetBulletInfo.

Output parameters:

AName – name of *bullet*[176]. It can also be read using GetItemText[310] method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using GetItemExtraIntPropertyEx[307] method.

ATag – *tag* of the item. Use SetItemTag[367] or SetBulletInfo[359] to change *tag* of item. This value can also be read using GetItemTag[310] method.

Additional output parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Additional item properties are returned by the methods GetItemExtraIntProperty[307] and GetItemExtraStrProperty[308].

See also methods:

- SetBulletInfo[359];
- GetItemStyle[310];
- GetItemExtraIntProperty[307];
- GetItemExtraStrProperty[308].

See also properties:

- ItemCount[244].

See also:

- Obtaining RichView items[116];
- Item types[90].
- *Tags*[96].

6.1.1.3.44 TRichView.GetCheckpointByNo

Returns the **No**-th *checkpoint*.

```
function GetCheckpointByNo(No: Integer): TCheckpointData[1055];
```

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[951])

See also methods:

- GetCheckpointNo[299];
- GetItemCheckpoint[305].

See also:

- "Checkpoints"[92].

6.1.1.3.45 TRichView.GetCheckpointInfo

Returns information about the *checkpoint* **CheckpointData**

```
procedure GetCheckpointInfo(CheckpointData: TCheckpointData[1055];
  out Tag: TRVTag[1096]; out Name: TRVUnicodeString[1099];
  out 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[379] event for this *checkpoint*.

See also properties:

- CPEventKind[233].

See also events:

- OnCheckpointVisible[379].

See also:

- "Checkpoints"[92];
- "Tags"[96].

6.1.1.3.46 TRichView.GetCheckpointItemNo

Returns index of the RichView item which owns the checkpoint **CheckpointData**

```
function GetCheckpointItemNo(CheckpointData: TCheckpointData[1055]): Integer;
```

Note: this method does not work with checkpoints in subdocuments (table cells[951])

See also:

- Item types[90];
- "Checkpoints"[92].

6.1.1.3.47 TRichView.GetCheckpointNo

Returns index of the *checkpoint* **CheckpointData**

```
function GetCheckpointNo(CheckpointData: TCheckpointData[1055]): 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[951])

See also methods:

- GetCheckpointByNo[298];
- GetCheckpointItemNo[299].

See also:

- "Checkpoints"[92].

6.1.1.3.48 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[1055];
  out X,Y: TRVCoord[1064]);
```

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[239]).

This method must be called only when the document is formatted.

See also methods:

- GetCheckpointY[300];
- GetCheckpointYEx[301];
- ScrollTo[875].

See also:

- "Checkpoints"[92];
- Scrolling[110].

6.1.1.3.49 TRichView.GetCheckpointY

Returns vertical coordinate of the **no**-th *checkpoint*

```
function GetCheckpointY(no: Integer): TRVCoord[1064];
```

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[301];
- GetCheckpointXY[300];
- ScrollTo[875].

See also:

- "Checkpoints"[92];
- Vertical scrolling[110].

6.1.1.3.50 TRichView.GetCheckpointYEx

Returns vertical coordinate of the *checkpoint* **CheckpointData**

```
function GetCheckpointYEx(
  CheckpointData: TCheckpointData[1055]): TRVCoord[1064];
```

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[300];
- GetCheckpointXY[300];
- ScrollTo[875].

See also:

- "Checkpoints"[92];
- Vertical scrolling[110].

6.1.1.3.51 TRichView.GetControlInfo

Returns main properties of the item of "inserted control" type[174]

```
procedure GetControlInfo(ItemNo: Integer;
  out AName: TRVUnicodeString[1099]; out Actrl: TControl;
  out AVAlign: TRVVAlign[1100]; out ATag: TRVTag[1096]);
```

(changes in version 18)

Input parameter:

ItemNo – index of the item. The item must be of "inserted control" type[174] (*rvsComponent*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetControlInfo.

Output parameters:

AName – name of control item[174]. Do not confuse with **Actrl**.Name property! This value can also be read using GetItemText[310] 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[367] or SetControlInfo[361] to change *tag* of item. This value can also be read using GetItemTag[310] method.

Additional item properties are returned by the methods GetItemExtraIntProperty[307] and GetItemExtraStrProperty[308].

See also methods:

- SetControlInfo[361];
- GetItemStyle[310];
- GetItemExtraIntProperty[307];
- GetItemExtraStrProperty[308].

See also properties:

- ItemCount[244].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.1.3.52 TRichView.GetFirstCheckpoint

Returns value identifying the first *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetFirstCheckpoint: TCheckpointData[1055];
```

GetFirstCheckpoint + GetNextCheckpoint[315] allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells[951]) are not included in *checkpoints* list of the document.

See also methods:

- GetNextCheckpoint[315], contains example;
- GetLastCheckpoint[313];
- GetPrevCheckpoint[319];

- GetCheckpointInfo[299];
- GetCheckpointXY[300].

See also:

- "Checkpoints"[92].

6.1.1.3.53 TRichView.GetFocusedItem

Returns the item having keyboard focus.

```
procedure GetFocusedItem(out ARVData: TCustomRVFormattedData[1014];
  out AItemNo: Integer);
```

There are two types of items which can have keyboard focus:

- hypertext[97] (text or pictures)
- inserted controls[174].

If hypertext item has focus, pressing **Enter** invokes OnJump[390] event for the focused link.

This method returns the focused item (the **AItemNo**-th item of **ARVData** object).

ARVData can be TRichView.RVData[254] or table cell[198].

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[469].

See also:

- TabNavigation[262] property.

6.1.1.3.54 TRichView.GetHotspotInfo

Returns main properties for item of *hotspot* type[178]

VCL and LCL:

```
procedure GetHotspotInfo(ItemNo: Integer; out AName: TRVUnicodeString[1099];
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag[1096]);
```

FireMonkey:

```
procedure GetHotspotInfo(ItemNo: Integer; out AName: TRVUnicodeString[1099];
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag[1096];
  out ImageWidth, ImageHeight: TRVStyleLength[1094]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of *hotspot*[178] type (*rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetHotspotInfo.

Output parameters:

AName – name of *hotspot*[178]. It can also be read using GetItemText[310] 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[307] 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[307] method.

ATag – *tag* of the item. Use SetItemTag[367] or SetHotspotInfo[364] to change *tag* of item. This value can also be read using GetItemTag[310] method.

Additional output parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Additional item properties are returned by the methods GetItemExtraIntProperty[307] and GetItemExtraStrProperty[308].

See also methods:

- SetHotspotInfo[364];
- GetItemStyle[310];
- GetItemExtraIntProperty[307];
- GetItemExtraStrProperty[308].

See also properties:

- ItemCount[244].

See also:

- Obtaining RichView items[116];
- Item types[90];
- *Tags*[96].

6.1.1.3.55 TRichView.GetItem

Returns object representing one item in document by its index **ItemNo** (zero-based)

```
function GetItem(ItemNo: Integer): TCustomRVItemInfo[899];
```

(introduced in version 1.4)

Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItem.

Do not free this object yourself.

Usually this method if used to get an item object for tables[179], labels[181], sequences[183], footnotes[185], endnotes[184], references to notes[190].

This method is inverse to GetItemNo[309].

See also methods:

- AddItem[276];
- GetItemStyle[310].

See also methods of TRichViewEdit:

- GetCurrentItem[517];
- GetCurrentItemEx[517].

See also properties:

- ItemCount[244].

See also events:

- OnItemAction[387].

See also:

- Obtaining RichView items[116].

6.1.1.3.56 TRichView.GetItemAt

Returns item [164] at the position **X**,**Y**.

```
function GetItemAt(X, Y: TRVCoord[1064];
  out RVData: TCustomRVFormattedData[1014];
  out 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 [254] or table cell [951] 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 [322].

See also events:

- OnRVMouseDown [402] (contains example);
- OnRVMouseUp [403].

Demo projects:

- Demos*\Assorted\PlanetQuiz\

6.1.1.3.57 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[1055];
```

Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItemCheckpoint.

See also methods:

- GetCheckpointInfo[299];
- GetCheckpointXY[300].

See also:

- "Checkpoints"[92].

6.1.1.3.58 TRichView.GetItemCoords

The methods return coordinates of the item in RichView document.

```
function GetItemCoords(ItemNo: Integer;
  out Left,Top: TRVCoord[1064]): Boolean;
function GetItemClientCoords(ItemNo: Integer;
  out Left,Top: TRVCoord[1064]): Boolean;
```

(introduced in version 1.4)

Input parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) 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[307] does not have these limitations.

Return value:

True if coordinates were successfully retrieved (document must be formatted)

See also:

- ClientToDocument, DocumentToClient[286].

See also methods of TCustomRVFormattedData

- GetOriginEx[1018].

6.1.1.3.59 TRichView.GetItemCoordsEx

Returns coordinates of the item in RichView document.

```
function GetItemCoordsEx(RVData: TCustomRVFormattedData[1014];
  ItemNo, Part: Integer; AllowFloating: Boolean;
  out R: TRVCoordRect[1064]): Boolean;
```

(introduced in version 14)

Input parameters:

RVData, ItemNo define the position of the item. **RVData** – an object containing this item (RichView.RVData[254] or table cell[951] 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[1100] 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[286].

The method must be called only when the document is formatted.

This method takes cell rotation[961] into account.

Return value:

True if coordinates were successfully retrieved (document must be formatted; and **Part** must be valid).

See also:

- GetItemCoords[306].

6.1.1.3.60 TRichView.GetItemExtraIntProperty -Ex

The methods return a value of the specified integer property for the **ItemNo**-th item.

```
function GetItemExtraIntProperty(ItemNo: Integer;
  Prop: TRVExtraItemProperty[1067]; out Value: Integer): Boolean;
```

```
function GetItemExtraIntPropertyEx(ItemNo, Prop: Integer;
  out Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

ItemNo – index of the item. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItemExtraIntProperty[Ex].

Value receives value of the property identified by **Prop**.

In **GetItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[1067]. 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[1119]

Return value:

True, if this item has this property. *False*, if not.

See also:

- SetItemExtraIntProperty[Ex][366];
- GetItemExtraStrProperty[308].

See also methods of TRichViewEdit:

- GetCurrentItemExtraIntProperty[Ex][518].

6.1.1.3.61 TRichView.GetItemExtraStrProperty -Ex

The methods return a value of the specified string property for the **ItemNo**-th item

```
function GetItemExtraStrProperty(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty[1072];
  out Value: TRVUnicodeString[1099]): Boolean;
function GetItemExtraStrPropertyEx(ItemNo, Prop: Integer;
  out Value: TRVUnicodeString[1099]): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

ItemNo – index of the item. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItemExtraStrProperty[Ex].

Value receives value of the property identified by **Prop**.

In **GetItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[1072]. 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[1124]

Return value:

True, if this item has this property. *False*, if not.

See also:

- SetItemExtraStrProperty[Ex] [367];
- GetItemExtra**Int**Property[Ex] [307].

See also methods of TRichViewEdit:

- GetCurrentItemExtraStrProperty[Ex] [519].

6.1.1.3.62 TRichView.GetItemNo

Returns index of item (zero-based).

```
function GetItemNo(Item: TCustomRVItemInfo[899]): Integer;
```

(introduced in version 1.4)

Item is an object representing the item. Usually this method if used to get index if table item [901].

This method is inverse to GetItem [304].

Returned value can be used, for example, in methods:

TRichViewEdit.BeginItemModify [503]/EndItemModify [511]

See also methods:

AddItem [276].

6.1.1.3.63 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 [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].GetItemPara.

Return value:

The returned value is an index in the collection of Style [260].ParaStyles [757]. 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 [325];
- IsFromNewLine [325].

See also:

- Paragraphs [100].

6.1.1.3.64 TRichView.GetItemStyle

Returns type (style) of the **ItemNo**-th item.

```
function GetItemStyle(ItemNo: Integer): Integer;
```

Parameters:

ItemNo – index of item (from 0 to ItemCount[244]-1). Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItemStyle.

Return value:

If the item is a text item[167], this function returns zero or positive value: index of text style of this item in the collection of styles (Style[260].TextStyles[764]).

If the item is not a text item, this function returns type of this item. It is a negative value, see rvs**** constants[1127].

See also properties of TRichViewEdit:

- CurItemStyle[481].

See also:

- Item types[90].

6.1.1.3.65 TRichView.GetItemTag

Returns *tag* of the **ItemNo**-th item

```
function GetItemTag(ItemNo: Integer): TRVTag[1096];
```

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetItemTag.

Use SetItemTag[367] to change tag of item.

See also methods:

- SetItemTag[367].

See also method of TRichViewEdit:

- GetCurrentTag[522].

See also:

- RichView items' "tags"[96].

6.1.1.3.66 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[1056];
function GetItemTextW(ItemNo: Integer): TRVUnicodeString[1099];
```

(Introduced in version 1.3, 1.7)

ItemNo – index of the item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].GetItemText*.

Unicode notes:

Internally, text is stored as Unicode. **GetItemTextA** converts Unicode to ANSI. For a text item, a code page for conversion is calculated basing on the its Charset [809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style [260].DefCodePage [745] is used.

GetItemText returns:

ANSI string, like **GetItemTextA**, in Delphi 2007 and older
Unicode (UTF-16) string, like **GetItemTextW**, in Delphi 2009 and newer
Unicode (UTF-8) string, in Lazarus [160]

See also:

- SetItemText -A -W [368].

See also method of TRichViewEdit:

- GetCurrentItemText -A -W [519].

See also:

- Obtaining Items of RichView [116];
- Unicode in RichView [135].

6.1.1.3.67 TRichView.GetItemVAlign

Returns vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
function GetItemVAlign(ItemNo: Integer): TRVVAlign[1100];
```

(introduced in version 12)

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount [244]-1. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].GetItemVAlign.

This method must not be called for text items [167], tables [179], *breaks* [173], tabs [168], list markers [180], footnotes [185] and endnotes [184].

Use SetItemVAlign [369] to change vertical alignment.

See also methods:

- SetItemVAlign [369].

See also method of TRichViewEdit:

- GetCurrentItemVAlign [520]

6.1.1.3.68 TRichView.GetJumpPointItemNo

Returns index of hypertext item (*hotspot*[178], *hot-picture*[172] or text hypertext) by its hypertext **id**.

```
function GetJumpPointItemNo(id: Integer): Integer;
```

Warning: do not use this function for document containing tables[196]. 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[312] instead.

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo[243].

Return value

Index of the hypertext item (in range 0..ItemCount[244]-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[469], if ReadOnly[488]=False, this function always returns -1 (except for calling it from inside OnWriteHyperlink[418], OnJump[390] or OnRVMouseMove[403]).

See also:

- RichView hypertext[97].

6.1.1.3.69 TRichView.GetJumpPointLocation

Returns information required to access item by its hypertext **id**.

```
procedure GetJumpPointLocation(id: Integer;
  out RVData: TCustomRVFormattedData[1014]; out ItemNo: Integer);
```

(introduced in version 1.4)

Input parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo[243].

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[254], table cell[951], 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[469], if ReadOnly[488]=False, this function always returns -1 (except for calling it from inside OnWriteHyperlink[418], OnJump[390] or OnRVMouseMove[403]).

See also:

- RichView hypertext[97].

6.1.1.3.70 TRichView.GetJumpPointY

Returns vertical coordinate of the hypertext link (*hotspot*[178], *hot-picture*[172] or text hypertext), identified by **id**

```
function GetJumpPointY(id: Integer): TRVCoord[1064];
```

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo[243].

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[469], if ReadOnly[488]=False, this function always returns 0 (except for calling it from inside OnJump[390] or OnRVMouseMove[403]).

See also methods:

- ScrollTo[875]

See also:

- RichView Hypertext[97]

6.1.1.3.71 TRichView.GetLastCheckpoint

Returns value identifying the last *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetLastCheckpoint: TCheckpointData[1055];
```

(introduced in version 1.5)

GetLastCheckpoint + GetPrevCheckpoint[319] 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[315], contains example;
- GetFirstCheckpoint[302];
- GetPrevCheckpoint[319];

- GetCheckpointInfo[299];
- GetCheckpointXY[300].

See also:

- "Checkpoints"[92].

6.1.1.3.72 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[244]-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[316](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[244]-1, GetOffsAfterItem[316](ItemCount[244]-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[310].

See also methods of TCustomRichViewEdit:

- GetCurrentLineCol[521].

6.1.1.3.73 TRichView.GetListMarkerInfo

Returns main properties of list marker[180] (paragraph bullet or numbering)

```
function GetListMarkerInfo(AItemNo: Integer;
  out AListNo, AListLevel, AStartFrom: Integer;
  out AUseStartFrom: Boolean): Integer;
```

(introduced in version 1.7)

Input parameter

AItemNo – index of the item, in range from 0 to ItemCount[244]-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[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].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[260].ListStyles[755]).

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[AListNo].Levels[842]).

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[369].

6.1.1.3.74 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[1055]): TCheckpointData[1055];
```

GetFirstCheckpoint[302] + **GetNextCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells[951]) are not included in *checkpoints* list of the document.

Example:

```
var Tag: TRVTag[1096];
    Name: String;
    CheckpointData: TCheckpointData;
```

```
    RaiseEvent: Boolean;
    s: TRVUnicodeString[1099];
begin
  MyListBox.Items.Clear;
  CheckpointData := MyRichView.GetFirstCheckpoint[302];
  while CheckpointData<>nil do
  begin
    MyRichView.GetCheckpointInfo[299](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[302];
- GetLastCheckpoint[313];
- GetPrevCheckpoint[319];
- GetCheckpointInfo[299];
- GetCheckpointXY[300].

See also:

- "Checkpoints"[92].

6.1.1.3.75 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[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetOffsAfterItem.

For text items, the method returns length of text + 1. The exception is empty text items formatted with style having EmptyWidth[810]>0; for them, the position after the item is 2.

For non-text items, it returns 1.

Example for TRichViewEdit[469]: moving caret to the end of document

```
var ItemNo, Offs: Integer;

  ItemNo := MyRichViewEdit.ItemCount-1;
  Offs := MyRichViewEdit.GetOffsAfterItem(ItemNo);
  MyRichViewEdit.SetSelectionBounds[372](ItemNo, Offs, ItemNo, Offs);
  MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds[372];
- GetOffsBeforeItem[317].

See also properties of TRichView:

- ItemCount[244].

See also properties of TRichViewEdit:

- CurItemNo[480];
- OffsetInCurItem[488].

See also:

- Working with selection[112];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[1133].

6.1.1.3.76 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[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetOffsBeforeItem.

For text items, the method returns 1 For non-text items, it returns 0.

Example for TRichViewEdit[469]: moving caret to the beginning of document

```
var ItemNo, Offs: Integer;

  ItemNo := 0;
  Offs := MyRichViewEdit.GetOffsBeforeItem(ItemNo);
  MyRichViewEdit.SetSelectionBounds[372](ItemNo, Offs, ItemNo, Offs);
  MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds[372];
- GetOffsAfterItem[316].

See also properties of TRichView:

- ItemCount[244].

See also properties of TRichViewEdit:

- CurItemNo[480];
- OffsetInCurItem[488].

See also:

- Working with selection[112];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[1133];

- Example how to move caret to the beginning of paragraph[602].

6.1.1.3.77 TRichView.GetPictureInfo

Returns main properties for item of picture[169] or *hot-picture*[172] type

```
procedure GetPictureInfo(ItemNo: Integer; out AName: TRVUnicodeString[1099];
  out Agr: TRVGraphic[1032]; out AVAlign: TRVVAlign[1100];
  out ATag: TRVTag[1096]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of picture[169] or *hot-picture*[172] type (*rvsPicture*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetPictureInfo.

Output parameters:

AName – name of the item. This value can also be read using GetItemText[310] 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[367] or SetPictureInfo[370] to change *tag* of item. This value can also be read using GetItemTag[310] 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[307] and GetItemExtraStrProperty[308].

See also methods:

- SetPictureInfo[370];
- GetItemStyle[310];
- GetItemExtraIntProperty[307];
- GetItemExtraStrProperty[308].

See also properties:

- ItemCount[244].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.1.3.78 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[1055]): TCheckpointData[1055];
```

(introduced in version 1.5)

GetLastCheckpoint[313] + **GetPrevCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells[951]) are not included in *checkpoints* list of document.

See also methods:

- GetFirstCheckpoint[302];
- GetLastCheckpoint[313];
- GetNextCheckpoint[315];

- GetCheckpointInfo[299];
- GetCheckpointXY[300].

See also:

- "Checkpoints"[92].

6.1.1.3.79 TRichView.GetRealDocumentPixelsPerInch

Returns the pixel depth (a number of pixels in a logical inch) used to draw documents.

```
function GetRealDocumentPixelsPerInch: Integer;
```

(introduced in version 18)

VCL and LCL

If DocumentPixelsPerInch[239] contains a positive value, this function returns it.

Otherwise, if RichViewPixelsPerInch[1115] (global variable) contains a positive value, this function returns it.

Otherwise, if the application supports multiple monitors having different pixel depth (Delphi 10.1+), this function returns the pixel depth of the monitor that contains a form containing this editor.

Otherwise, this function returns Screen.PixelsPerInch (a pixel depth of the primary monitor).

FireMonkey

Returns 96 * ZoomPercent[265]/100.

6.1.1.3.80 TRichView.GetSelectedImage

Returns the selected image

```
function GetSelectedImage: TRVGraphic[1032];
```

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 TRichView, 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 [112].

6.1.1.3.81 TRichView.GetSelectionBounds

Returns bounds of selected part of document.

```
procedure GetSelectionBounds(out StartItemNo, StartItemOffs,
  EndItemNo, EndItemOffs: Integer; Normalize: Boolean);
```

Parameters

StartItemNo – index of the first selected item.

StartItemOffs:

- if the first item is a text item [167], 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 [810]>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 [810]>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[1047] unit instead.

See also methods:

- SetSelectionBounds[372].

See also:

- Working with selection.[112]

6.1.1.3.82 TRichView.GetSelText -A -W

These methods return the selected part of the document as a text.

```
function GetSelText: String;
function GetSelTextA: TRVAnsiString[1056];
function GetSelTextW: TRVUnicodeString[1099];
```

If there is nothing selected, these methods return '' (empty string)

Unicode notes:

Internally, text is stored as Unicode. **GetSelTextA** converts Unicode to ANSI using Style[260].DefCodePage[745] code page.

GetSelText returns:

- ANSI string, like **GetSelTextA**, in Delphi 2007 and older
- Unicode (UTF-16) string, like **GetSelTextW**, in Delphi 2009 and newer
- Unicode (UTF-8) string, in Lazarus[160]

These methods must be called only when the document is formatted.

See also:

- Working with selection[112].

6.1.1.3.83 TRichView.GetTextInfo

Returns text and *tag* for the **ItemNo**-th item

```
procedure GetTextInfo(ItemNo: Integer; out AText: TRVUnicodeString[1099];
  out ATag: TRVTag[1096]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of text type (GetItemStyle[310] must return zero or positive value for this item), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].GetTextInfo.

Output parameters:

AText – item text (visible text). This is the same value as returned by GetItemText[310].

ATag – *tag* of the item. Use SetItemTag[367] to change *tag* of item. The *tag* can also be returned by GetItemTag[310].

See also methods:

- GetItemStyle[310];
- GetItemText[310], SetItemText[368];
- GetItemTag[310], SetItemTag[367].

See also properties:

- ItemCount[244].

See also:

- Obtaining items[116];
- Item types[90];
- "Tags"[96].

6.1.1.3.84 TRichView.GetWordAt -A -W

Returns word at the specified coordinates

```
procedure GetWordAt(X,Y: TRVCoord[1064];
  out RVData: TCustomRVFormattedData[1014]; out ItemNo: Integer;
  out Word: TRVUnicodeString[1099]); overload;
function GetWordAtA(X,Y: TRVCoord[1064]): TRVAnsiString[1056];
function GetWordAtW(X,Y: TRVCoord[1064]): TRVUnicodeString[1099]
function GetWordAt(X,Y: TRVCoord[1064]): 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[167]), 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[235] 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[260].DefCodePage[745].

GetWordAt returns:

- ANSI string, like **GetWordAtA**, in Delphi 2007 and older
- Unicode (UTF-16) string, like **GetWordAtW**, in Delphi 2009 and newer

- Unicode (UTF-8) string, in Lazarus [160]

For example, you can use this method inside OnRVMouseDown [402] and OnRVMouseUp [403].

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: TRVCoord[1064]);
var
  LItemNo: Integer;
  LRVData: TCustomRVFormattedData;
  LWord: TRVUnicodeString[1099];
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[310](LItemNo)>=0 then // text item?
    Label1.Caption := LWord;
end;
```

See also events:

- OnRVMouseDown [402];
- OnRVMouseUp [403];
- OnRVRightClick [404];
- OnRVDblClick [399].

See also properties:

- Delimiters [235].

See also methods:

- SelectWordAt [356]

6.1.1.3.85 TRichView.InsertRVFFromStream

Inserts RVF content from **Stream** at 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 [244]. Items of subdocuments (table cells [951]) 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[224].

If style templates are used[263], and RVFTextStylesReadMode[258]=RVFParaStylesReadMode[258] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style[260].StyleTemplates[762], and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] 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[244]);
  Stream.Free;
end;
  ...
  RichView1.Clear[286];
  RichView1.DeleteUnusedStyles[293](True, True, True);
  AddDoc('c:\file1.rvf', RichView1);
  AddDoc('c:\file2.rvf', RichView1);
  AddDoc('c:\file3.rvf', RichView1);
  RichView1.Format[295];
```

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[244];
  Stream := TFileStream.Create(FileName, fmOpenRead);
  rv.InsertRVFFromStream(Stream, rv.ItemCount[244]);
  Stream.Free;
  if (ItemCount>0) and (ItemCount<rv.ItemCount[244]) then
    rv.PageBreaksBeforeItems[250][ItemCount] := True;
end;
```

See also properties:

- RVFOptions[254];
- RVFWarnings[258];
- RVFTextStylesReadMode[258] (this method type: RVF insertion);
- RVFParaStylesReadMode[258] (this method type: RVF insertion).

See also events:

- OnRVFImageListNeeded[400];
- OnRVFControlNeeded[400];
- OnRVFPictureNeeded[401];
- OnControlAction[380] (ControlAction=*rvcaAfterRVFLoad*);

- OnStyleTemplatesChange [417].

See also methods:

- AppendRVFFromStream [283];
- LoadRVFFromStream [337];
- LoadRVF [336];
- Format [295];
- FormatTail [296].

See also methods of TRichViewEdit:

- InsertRVFFromStreamEd [538].
- **See also:**
- TRichView component editor [224];
- Saving and loading [127];
- RVF overview [129].

6.1.1.3.86 TRichView.IsFromNewLine

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 [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].IsFromNewLine.

This method returns *True* both in cases when this item starts a new paragraph (see IsParaStart [325]) 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 [357](*False*)).

See also methods:

- IsParaStart [325];
- GetItemPara [309].

See also:

- Paragraphs [100].

6.1.1.3.87 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 [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].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 [357](*False*)).

See also methods:

- IsFromNewLine [325];
- GetItemPara [309].

See also:

- Paragraphs [100];
- Example how to move caret to the beginning of paragraph [602].

6.1.1.3.88 TRichView.LiveSpellingValidateWord

Removes live spelling underlines from all occurrences of **AWord**.

```
procedure LiveSpellingValidateWord(const AWord: TRVUnicodeString[1099]);
```

(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.

If this TRichView control is registered [736] in TRVSpellChecker [728] component, the spell-checker calls this method automatically.

See also:

- Live spelling check in RichView [138].

6.1.1.3.89 TRichView.LoadDocX

Appends the content of DocX (Microsoft Word Document) file **FileName** to the document.

```
function LoadDocX(const FileName: TRVUnicodeString[1099]):Boolean;
```

(Introduced in version 19)

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 [286] method before loading (and maybe DeleteUnusedStyles [293] after Clear)

Method type: viewer-style.

Parameters for loading are in RTFReadProperties [253]. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode [466] and ParaStyleMode [464]. The imported DocX may look absolutely different depending on values of these properties! Setting for DocX loading can be changed in the TRichView component editor [224].

See important note on loading external images from DocX [1137].

By default, hyperlink targets are loaded in items tags [96]. You can customize loading using OnReadHyperlink [395] event.

If style templates are used [263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode [260]<>*rvstimIgnoreSourceStyleTemplates*, the method loads a DocX style sheet into Style [260].StyleTemplates [762]. If TRichView was completely empty before loading, **LoadDocX** replaces the existing style templates. Otherwise, it merges the DocX style sheet into the style templates, and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

Return value: "Was reading successful?" (extended information is in RTFReadProperties.DocXErrorCode[462])

See also methods of TRichView:

- LoadDocXFromStream[327];
- SaveDocX[341].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417];
- OnProgress[392];
- OnAssignImageFileName[379].

See also methods of TRichViewEdit:

- InsertDocXFromFileEd[526].

See also:

- Saving and loading in RichView[127].

6.1.1.3.90 TRichView.LoadDocXFromStream

Appends the content of DocX (Microsoft Word Document) stream **Stream** to the document.

```
function LoadRTFFromStream(Stream: TStream):Boolean;
```

(Introduced in version 19)

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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear)

Method type: viewer-style.

Parameters for loading are in TRichView.RTFReadProperties[253]. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. The imported DocX may look absolutely different depending on values of these properties! Setting for DocX loading can be changed in the TRichView component editor[224].

See important note on loading external images from RTF[1137].

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method loads a DocX style sheet into Style[260].StyleTemplates[762]. If TRichView was completely empty before loading, **LoadDocXFromStream** replaces the existing style templates. Otherwise, it merges the DocX style sheet into the style templates, and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

Return value: "Was reading successful?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode[462])

See also methods of TRichView:

- LoadDocX[326];
- SaveDocXToStream[342].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417];
- OnProgress[392].

See also methods of TRichViewEdit:

- InsertDocXFromStreamEd[527].

See also:

- Saving and loading in RichView[127].

6.1.1.3.91 TRichView.LoadFromFile -Ex

The methods load a document from file **FileName**, autodetecting its format (RVF[129] /RTF/HTML/DocX/Markdown/text)

```
function LoadFromFile(const FileName: TRVUnicodeString[1099];
  IsTextUnicode: TRVYesNoAuto[1101];
  AllowMarkdown: Boolean = False;
  CodePage: TRVCodePage[1062] = RV_CP_DEFAULT[1118]): Boolean;): Boolean;
function LoadFromFileEx(const FileName: TRVUnicodeString[1099];
  IsTextUnicode: TRVYesNoAuto[1101]; AllowedFormats: TRVLoadFormats;
  out Unicode: Boolean;
  out LoadedFormat: TRVLoadFormat;
  CodePage: TRVCodePage[1062] = RV_CP_DEFAULT[1118]): Boolean;): Boolean;
```

(introduced in version 21, modified in version 23)

These methods are analogs of LoadFromStream and LoadFromStreamEx[329].

Methods type: viewer-style.

Return value:

"Was reading successful?"

See also methods of TRichView:

- LoadRVF[336];
- LoadRTF[334];
- LoadDocX[326];
- LoadMarkdown[332];
- LoadHTML[330];
- LoadText, LoadTextW[338].

6.1.1.3.92 TRichView.LoadFromStream -Ex

The methods load a document from **Stream**, autodetecting its format (RVF[129] /RTF/HTML/DocX/Markdown/text)

```
function LoadFromStream(Stream: TStream; IsTextUnicode: TRVYesNoAuto[1101];
  AllowMarkdown: Boolean = False;
  const Path: TRVUnicodeString[1099] = '';
  CodePage: TRVCodePage[1062] = RV_CP_DEFAULT[1118]): Boolean;
function LoadFromStreamEx(Stream: TStream;
  IsTextUnicode: TRVYesNoAuto[1101]; AllowedFormats: TRVLoadFormats[1081];
  out Unicode: Boolean; out LoadedFormat: TRVLoadFormat[1081];
  const Path: TRVUnicodeString = '';
  CodePage: TRVCodePage[1062] = RV_CP_DEFAULT[1118]): Boolean;): Boolean;
```

(introduced in version 10, modified in version 21 and 23)

These methods can detect both normal RTF, and RTF converted to Unicode (UTF-16) string.

HTML is detected by the presence of '<html' or '<head' within the leading 1024 characters.

Text or Markdown is loaded if the stream does not contain RVF, RTF, HTML or DocX. The methods do not load text or Markdown if **Stream** contains characters having codes less than ord(' ').

Path is the base path for images that may be referred from a document.

IsTextUnicode specifies whether the text/Markdown is Unicode[135] (UTF-16):

- *rvynaNo* – not Unicode (i.e, not UTF-16);
- *rvynaYes* – Unicode (UTF-16);
- *rvynaAuto* – autodetecting ANSI/Unicode (a wrong detection is possible).

If Unicode is specified or detected, a text or a Markdown is loaded using UTF-16 encoding.

If Unicode is not specified or detected, a text and Markdown is loaded using the code page specified in the **CodePage** parameter. The default value of **CodePage** = RV_CP_DEFAULT[1118] (meaning UTF-8 for Markdown, default ANSI code page for plain text in Windows, UTF-8 for plain text in non-Windows).

LoadFromStream: if **AllowMarkdown** = *True*, a Markdown is loaded; otherwise, a plain text is loaded.

LoadFromStreamEx: allowed formats are listed in **AllowedFormats**. If it contains both *rvlfText* and *rvlfMarkdown*, Markdown is loaded. The loaded format is returned in **LoadedFormat**. If the loaded text/markdown/HTML/RTF is stored as UTF-16 or UTF-8, **Unicode** = *True* on exit.

Methods type: viewer-style.

Return value:

"Was reading successful?"

See also methods of TRichView:

- LoadFromFile, LoadFromFileEx[328]
- LoadRVFFromStream[337];
- LoadRTFFromStream[335];
- LoadDocXFromStream[327];

- LoadMarkdownFromStream[333];
- LoadHTMLFromStream[332];
- LoadTextFromStream, LoadTextFromStreamW[339].

6.1.1.3.93 TRichView.LoadHTML

Appends the content of HTML file **FileName** to the document.

```
function LoadHTML(const FileName: TRVUnicodeString[1099];
  CodePage: TRVCodePage[1062] = CP_ACP): Boolean;
```

(introduced in version 21)

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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear)

Parameters

FileName – the name of the input HTML file.

CodePage defines encoding of HTML file. If it is equal to 0 (CP_ACP), encoding is detected by TRichView.

Method type: viewer-style.

Parameters for loading are in HTMLReadProperties[243].

See important note on loading external images from HTML[1137].

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

Encoding detection

TRichView detects HTML encoding by Unicode byte order mark (UTF-8, UTF-16 little endian, UTF-16 big endian) and by charset specified in HTML. Unicode byte order marks have higher priority than charset specified in HTML.

Without Unicode byte order mark, TRichView cannot detect UTF-16, even if it is specified in HTML. To load UTF-16 without byte order mark, you need to specify in **Charset** parameter of this method:

- 1200 for UTF-16 little endian (default Unicode format in Windows)
- 1201 for UTF-16 big endian.

Alternatively, you can use LoadFromFile[328] method, it can detect and load UTF-16 HTML.

Style templates

If style templates are used[263], and StyleTemplateInsertMode[260] <>*rvstimIgnoreSourceStyleTemplates*, the method loads an HTML style sheet into Style[260].StyleTemplates[762]. If TRichView was completely empty before loading, **LoadHTML** replaces the existing style templates. Otherwise, it merges the HTML style sheet into the style templates, and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

HTML style sheet is created basing on the properties defined for certain HTML tag after reading the starting tag of <body>.

The table bellow shows HTML tags and corresponding style templates.

HTML Tag	StyleTemplate Name	StyleTemplate Kind
p	'Normal'	*rvstkPara*
h1..h6	'heading 1' .. 'heading 6'	*rvstkParaText*
pre	'HTML Preformatted'	
address	'HTML Address'	
blockquote	'Quote'	
strong	'Strong'	*rvstkText*
em	'Emphasis'	
a	'Hyperlink'	
code	'HTML Code'	
acronym	'HTML Acronym'	
var	'HTML Variable'	
kbd	'HTML Keyboard'	
samp	'HTML Sample'	
tt	'HTML Typewriter'	
dfn	'HTML Definition'	
cite	'HTML Cite'	

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadHTMLFromStream 332;
- SaveHTML 342.

See also events of TRichView:

- OnImportPicture 386;
- OnImportFile 385;
- OnReadHyperlink 395;
- OnStyleTemplatesChange 417;
- OnProgress 392;
- OnAssignImageFileName 379.

See also methods of TRichViewEdit:

- InsertHTMLFromFileEd 529.

See also:

- Saving and loading in TRichView 127.

6.1.1.3.94 TRichView.LoadHTMLFromStream

Appends the content of HTML from **Stream**.

```
function LoadHTMLFromStream(Stream: TStream;
  const Path: TRVUnicodeString[1099] = '';
  CodePage: TRVCodePage[1062] = CP_ACP): Boolean;
```

(introduced in version 21)

The same as LoadHTML[330], but loads from a stream instead of a file.

Parameters

Stream – stream containing HTML

Path defines a base path for images. If HTMLReadProperties[243].BasePathLinks[433] = *True*, this path is also used for hyperlinks.

CodePage defines encoding of HTML file. If it is equal to 0 (CP_ACP), encoding is detected by TRichView.

Method type: viewer-style.

Parameters for loading are in HTMLReadProperties[243].

See details in the topic about LoadHTML[330].

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadHTML[330];
- SaveHTMLToStream[346].

See also events of TRichView:

- OnImportPicture[386];
- OnImportFile[385];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417];
- OnProgress[392];
- OnAssignImageFileName[379].

See also methods of TRichViewEdit:

- InsertHTMLFromStreamEd[530].

See also:

- Saving and loading in TRichView[127].

6.1.1.3.95 TRichView.LoadMarkdown

Appends the content of ***Markdown*** file **FileName** to the document.

```
function LoadMarkdown(const FileName: TRVUnicodeString[1099];
  CodePage: TRVCodePage[1062] = CP_UTF8): Boolean;
```

(Introduced in version 20)

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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear).

By default, the method assumes that the file has UTF-8 encoding; but you can specify another **CodePage** in the parameter.

Method type: viewer-style.

Parameters for loading are in MarkdownProperties[245]. See the topic about the class of this property[447] for the detailed explanation of Markdown import and export.

Unlike other loading methods, Markdown loading does not have an option to restrict formatting to existing text, paragraphs and list styles: it may add new styles.

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadMarkdownFromStream[333];
- SaveMarkdown[348].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnAssignImageFileName[379].

See also methods of TRichViewEdit:

- InsertMarkdownFromFileEd[532].

See also:

- Saving and loading in TRichView[127].

6.1.1.3.96 TRichView.LoadMarkdownFromStream

Appends the content of ***Markdown*** stream **Stream** to the document.

```
function LoadMarkdownFromStream(Stream: TStream;
  const Path: TRVUnicodeString[1099];
  CodePage: TRVCodePage = CP_UTF8[1062]): Boolean;
```

(Introduced in version 20)

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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear).

By default, the method assumes that the stream has UTF-8 encoding; but you can specify another **CodePage** in the parameter.

Path defines a base path for images. If *rvmdloBasePathLinks* is included MarkdownProperties[245].LoadOptions[453], this path is also used for hyperlinks.

Method type: viewer-style.

Parameters for loading are in MarkdownProperties[245]. See the topic about the class of this property[447] for the detailed explanation of Markdown import and export.

Unlike other loading methods, Markdown loading does not have an option to restrict formatting to existing text, paragraphs and list styles: it may add new styles.

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadMarkdown[332];
- SaveMarkdown[348].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnAssignImageFileName[379].

See also methods of TRichViewEdit:

- InsertMarkdownFromStreamEd[532].

See also:

- Saving and loading in TRichView[127].

6.1.1.3.97 TRichView.LoadRTF

Appends the content of RTF (Rich Text Format) file **FileName** to the document.

```
function LoadRTF(const FileName: TRVUnicodeString[1099]):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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear)

Method type: viewer-style.

Parameters for loading are in RTFReadProperties[253]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. 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[224].

See important note on loading external images from RTF[1137].

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method loads an RTF style sheet into Style[260].StyleTemplates[762]. 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[260]. The method calls OnStyleTemplatesChange[417] event.

Return value: "Was reading successful?" (extended information is in RTFReadProperties.RTFErrorCode[466])

See also methods of TRichView:

- LoadRTFFromStream[335];
- SaveRTF[350].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417];
- OnProgress[392];
- OnAssignImageFileName[379].

See also methods of TRichViewEdit:

- InsertRTFFromFileEd[536].

See also:

- Saving and loading in RichView[127].

6.1.1.3.98 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[286] method before loading (and maybe DeleteUnusedStyles[293] after Clear)

Method type: viewer-style.

Parameters for loading are in TRichView.RTFReadProperties[253]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. 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[224].

See important note on loading external images from RTF[1137].

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method loads an RTF style sheet into Style[260].StyleTemplates[762]. 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[260]. The method calls OnStyleTemplatesChange[417] event.

Return value: "Was reading successful?" (extended information is in TRichView.RTFReadProperties.RTFErrorCode[466])

See also methods of TRichView:

- LoadRTF[334];
- SaveRTFToStream[351].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417];
- OnProgress[392].

See also methods of TRichViewEdit:

- InsertRTFFromStreamEd[536].

See also:

- Saving and loading in RichView[127].

6.1.1.3.99 TRichView.LoadRVF

Clears the document and loads a new document from the file **FileName** as RVF (RichView Format[129]).

```
function LoadRVF(const FileName: TRVUnicodeString[1099]): Boolean;
```

(changed in version 18)

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor[224].

Return value: "Was reading successful?"

See also properties:

- RVFOptions[254];
- RVFWarnings[258];
- RVFTextStylesReadMode[258] (this method type: RVF loading);
- RVFParaStylesReadMode[258] (this method type: RVF loading);
- UseStyleTemplates[263].

See also events:

- OnRVFImageListNeeded[400];
- OnRVFControlNeeded[400];
- OnRVFPictureNeeded[401];
- OnControlAction[380] (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange[417];
- OnProgress[392].

See also methods:

- LoadRVFFromStream[337];
- AppendRVFFromStream[283];
- InsertRVFFromStream[323];

- Format [295];
- FormatTail [296].

See also methods of TRichViewEdit:

- InsertRVFFromFileEd [537].

- **See also:**

- TRichView component editor [224];
- Saving and loading [127];
- RVF overview [129].

6.1.1.3.100 TRichView.LoadRVFFromStream

Clears the document and loads a new document from the file stream **Stream** as RVF (RichView Format [129]).

```
function LoadRVFFromStream(Stream: TStream): Boolean;
```

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor [224].

Return value: "Was reading successful?"

See also properties:

- RVFOptions [254];
- RVFWarnings [258];
- RVFTextStylesReadMode [258] (this method type: RVF loading);
- RVFParaStylesReadMode [258] (this method type: RVF loading);
- UseStyleTemplates [263].

See also events:

- OnRVFImageListNeeded [400];
- OnRVFControlNeeded [400];
- OnRVFPictureNeeded [401];
- OnControlAction [380] (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange [417];
- OnProgress [392].

See also methods:

- LoadRVF [336];
- AppendRVFFromStream [283];
- InsertRVFFromStream [323];
- Format [295];
- FormatTail [296].

See also methods of TRichViewEdit:

- InsertRVFFromFileEd [537].

- **See also:**

- TRichView component editor [224];
- Saving and loading [127];
- RVF overview [129].

6.1.1.3.101 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[1099];
  StyleNo, ParaNo: Integer;  AsSingleParagraph: Boolean;
  CodePage: Cardinal=CP_ACP):Boolean;

function LoadTextW(const FileName: TRVUnicodeString[1099];
  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[286] method before loading (and may be DeleteUnusedStyles[293] after Clear).

Parameters

StyleNo – an index of text style[764] which will be used for new text. It defines font attributes for the added text.

ParaNo – an index of paragraph style[757] 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[357]);
- *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[263] 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[339];
- AddTextNLA, AddTextNLW[281];
- Format[295];
- FormatTail[296];
- SetAddParagraphMode[357];
- SaveText, SaveTextW[352].

See also methods of TRichViewEdit:

- InsertTextFromFile, InsertTextFromFileW [541].

See also properties of TRVStyle:

- TextStyles [764];
- SpacesInTab [761].

See also:

- Other methods for appending items [107];
- Saving and loading [127];
- Unicode in RichView [135];
- Example how to load UTF-8 files [468].

6.1.1.3.102 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 [338], but read text from streams instead of files.

6.1.1.3.103 TRichView.MarkStylesInUse

Stores lists of used text [764], paragraph [757] and list [755] styles.

```
procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData[1031]);
```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents [129] (but stored in file or in the Registry). In this case, several documents share the same collections of styles. See TRVDeleteUnusedStylesData [1031] 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 [293] instead.

See also methods:

- DeleteMarkedStyles [292];
- DeleteUnusedStyles [293].

6.1.1.3.104 TRichView.Reformat

Reformats the document.

Only formatted documents can be displayed or edited.

```
procedure Reformat;
```

(introduced in v1.9)

This method works like Format [295], but keeps the selection (and caret position in TRichViewEdit [469]).

Document must be formatted before calling this method.

This method is useful, for example, if you want to reformat document after changing margins.

6.1.1.3.105 TRichView.RefreshListMarkers

Recalculates list markers [180]

```
procedure RefreshListMarkers;
```

(introduced in v1.8)

Call it (before calling Format [295]) if you changed ListType [864] or FormatString [861] of list style level of marker that is already inserted in TRichView. It's usually not necessary to use this method.

6.1.1.3.106 TRichView.RefreshListSequences

Recalculates numbered sequences [183], footnotes [185], endnotes [184], and sidenotes [186].

```
procedure RefreshSequences;
```

(introduced in v13)

Call it (before calling Format [295]) if you changed FootnoteNumbering [746], EndnoteNumbering [746] or SidenoteNumbering [746] of TRVStyle [739] component linked [260] to this TRichView control, or a property of a numbered sequence item that was already inserted, to update existing document containing footnotes or endnotes.

6.1.1.3.107 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 [244]-1. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].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 [247].

See also methods of TRichView:

- GetItemCheckpoint [305];
- SetCheckpointInfo [360].

See also properties:

- ItemCount [244].

See also methods of RichViewEdit:

- RemoveCheckpointEd [550];
- RemoveCurrentCheckpoint [551].

See also:

- "Checkpoints [92]";
- Modifying RichView items [117].

6.1.1.3.108 TRichView.ResetAnimation

Rewinds all inserted animated pictures to the first frame.

```
procedure ResetAnimation;
```

(introduced in version 10)

See also methods:

- StartAnimation [374];
- StopAnimation [375].

See also properties:

- AnimationMode [228].

See also:

- Animation in TRichView [124];
- Pictures [169];
- *Hot-Pictures* [172].

6.1.1.3.109 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[1099];
  SelectionOnly: Boolean):Boolean;
```

(introduced in version 15; changed in version 18)

This method is available for Delphi 5+, C++Builder 2006+

The method uses *rvrtfSaveDocParameters* and *rvrtfSaveHeaderFooter* options from RTFOptions [252] property.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile [405] event.

By default, items tags [96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink [418] event.

You can save additional information about non-text objects in OnWriteObjectProperties [420] event.

You can:

- completely change DocX code for certain items using OnSaveItemToFile [411] event;
- insert OOXML codes [814] in text;
- insert additional code in DocX in OnSaveDocXExtra [407] event.

Return value: "successful saving?"

See also methods:

- SaveDocXToStream [342];
- LoadDocX [326]

See also properties:

- RTFOptions [252];
- UseStyleTemplates [263].

See also events:

- OnWriteHyperlink [418];
- OnWriteObjectProperties [420];
- OnSaveComponentToFile [405];
- OnSaveDocXExtra [407];
- OnSaveItemToFile [411];
- OnProgress [392].

See also global variables and constants:

- RichViewSaveHyperlinksInDocXAsFields [1116]

See also:

- Saving and loading in TRichView [127].

6.1.1.3.110 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 5+, C++Builder 2006+

The same as SaveDocX [341], but saves to a stream instead of a file.

See also

- LoadDocXFromStream [327]

6.1.1.3.111 TRichView.SaveHTML

Exports TRichView document to HTML (or XHTML) file

```
function SaveHTML(const FileName: TRVUnicodeString [1099];
  Part: TRVHTMLSavingPart [1079] = rvhtmlspComplete): Boolean;
  overload;
```

(introduced in version 21)

This method saves HTML file and optionally a set of images (in separate files).

Parameters:

FileName – the name of the output HTML file.

Part – the part of HTML to save. By default, the whole HTML is saved.

The following properties are used:

- HTMLSaveProperties [244]
- DocParameters [237].Title [427], Author [424], and Comments [424].

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile[405] event.

By default, items tags[96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink[418] event.

You can save additional information about non-text objects in OnWriteObjectProperties[420] event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile[411] event;
- insert additional code in HTML in OnSaveHTMLExtra[409] and OnSaveParaToHTML[413] events;
- change how images are saved using OnHTMLSaveImage[383] or OnSaveImage2[410] event,
- insert HTML codes[814] in text.

Return value: "successful saving?"

See also methods:

- SaveHTMLToStream[346]

See also events:

- OnWriteHyperlink[418];
- OnWriteObjectProperties[420];
- OnSaveComponentToFile[405];
- OnSaveHTMLExtra[409];
- OnHTMLSaveImage[383];
- OnSaveImage2[410];
- OnSaveItemToFile[411];
- OnSaveParaToHTML[413];
- OnProgress[392].

See also:

- Export to HTML[132];
- RichViewSavePInHTML[1107] typed constant;
- TFontInfo.Options[821] (HTML codes);
- Saving and loading[127].

6.1.1.3.112 TRichView.SaveHTML (deprecated)

Deprecated, use the new version of SaveHTML[342].

Exports TRichView document to HTML or XHTML file, using HTML tags like <font>, <b>, , etc.

```
function SaveHTML(const FileName, Title,
  ImagesPrefix: TRVUnicodeString[1099];
  Options: TRVSaveOptions[1086]): Boolean; overload;
```

(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[345] 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[1086] for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile[405] event.

By default, items tags[96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink[418] event.

You can save additional information about non-text objects in OnWriteObjectProperties[420] event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile[411] event;
- insert additional code in HTML in OnSaveHTMLExtra[409] and OnSaveParaToHTML[413] events;
- change how images are saved using OnHTMLSaveImage[383] or OnSaveImage2[410] event,
- insert HTML codes[814] in text.

Return value: "successful saving?"

See also methods:

- SaveHTMLEx[345];
- SaveHTMLToStream[347].

See also events:

- OnWriteHyperlink[418];
- OnWriteObjectProperties[420];
- OnSaveComponentToFile[405];
- OnSaveHTMLExtra[409];
- OnHTMLSaveImage[383];
- OnSaveImage2[410];
- OnSaveItemToFile[411];
- OnSaveParaToHTML[413];
- OnProgress[392].

See also:

- Export to HTML[132];
- RichViewSavePInHTML[1107] typed constant;
- TFontInfo.Options[821] (HTML codes);
- Saving and loading[127].

6.1.1.3.113 TRichView.SaveHTMLEx (deprecated)

Deprecated, use SaveHTML [342]

Exports TRichView document to HTML or XHTML file, using CSS (Cascading Style Sheets)

```
function SaveHTMLEx(const FileName, Title, ImagesPrefix, ExtraStyles,
  ExternalCSS, CPPrefix: TRVUnicodeString[1099];
  Options: TRVSaveOptions[1086]): 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[772]) instead of saving CSS into HTML file).

CPPrefix –- first part of *checkpoints*[92] 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[1086] for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile[405] event.

By default, items tags[96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink[418] event.

You can save additional information about non-text objects in OnWriteObjectProperties[420] event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile[411] event;
- insert additional code in HTML in OnSaveHTMLExtra[409] and OnSaveParaToHTML[413] events;
- change how images are saved using OnHTMLSaveImage[383] or OnSaveImage2[410] event,
- insert HTML codes[814] 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[809] 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[343];
- SaveHTMLToStreamEx[348].

See also events:

- OnWriteHyperlink[418];
- OnWriteObjectProperties[420];
- OnSaveComponentToFile[405];
- OnSaveHTMLExtra[409];
- OnHTMLSaveImage[383];
- OnSaveImage2[410];
- OnSaveItemToFile[411];
- OnSaveParaToHTML[413];
- OnProgress[392].

See also:

- Export to HTML[132];
- TFontInfo.Options[821] (HTML codes);
- RichViewSaveDivInHTMLEx[1107] typed constant;
- Saving and loading[127].

6.1.1.3.114 TRichView.SaveHTMLToStream

Exports TRichView document to HTML (or XHTML) stream.

```
function SaveHTMLToStream(Stream: TStream;
  const Path: TRVUnicodeString[1099] = '';
  Part: TRVHTMLSavingPart[1079] = rvhtmlspComplete): Boolean;
  overload;
```

(introduced in version 21)

This method saves HTML file and optionally a set of images (in separate files).

Parameters:

Stream – stream for HTML saving

Path – path where to save images (must have '\' as the last character).

Part – the part of HTML to save. By default, the whole HTML is saved.

The following properties are used:

- HTMLSaveProperties[244]
- DocParameters[237].Title[427], Author[424], and Comments[424].

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile[405] event.

By default, items tags[96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink[418] event.

You can save additional information about non-text objects in OnWriteObjectProperties[420] event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile[411] event;
- insert additional code in HTML in OnSaveHTMLExtra[409] and OnSaveParaToHTML[413] events;
- change how images are saved using OnHTMLSaveImage[383] or OnSaveImage2[410] event,
- insert HTML codes[814] in text.

Return value: "successful saving?"

See also methods:

- SaveHTML[342]

See also events:

- OnWriteHyperlink[418];
- OnWriteObjectProperties[420];
- OnSaveComponentToFile[405];
- OnSaveHTMLExtra[409];
- OnHTMLSaveImage[383];
- OnSaveImage2[410];
- OnSaveItemToFile[411];
- OnSaveParaToHTML[413];
- OnProgress[392].

See also:

- Export to HTML[132];
- RichViewSavePInHTML[1107] typed constant;
- TFontInfo.Options[821] (HTML codes);
- Saving and loading[127].

6.1.1.3.115 TRichView.SaveHTMLToStream (deprecated)

Deprecated, use the new version of SaveHTMLToStream[346].

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[1099];
  Options: TRVSaveOptions[1086]): Boolean; overload;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTML[343], 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[343].

6.1.1.3.116 TRichView.SaveHTMLToStreamEx (deprecated)

Deprecated, use SaveHTMLToStream[346].

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[1099];
  Options: TRVSaveOptions[1086]): Boolean;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTMLEx[345], 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[345].

6.1.1.3.117 TRichView.SaveMarkdown

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the file **FileName** as ***Markdown***.

```
function SaveMarkdown(const FileName: TRVUnicodeString[1099];
  SelectionOnly: Boolean;
  CodePage: TRVCodePage[1062] = CP_UTF8): Boolean;
```

(introduced in version 19, changed in verison 23)

Parameters

FileName – output file name;

SelectionOnly – if *True*, only selected part of the document is saved.

CodePage – text encoding (UTF-8 by default)

Return value: "successful saving?"

See also properties:

- MarkdownProperties[245].

See also methods:

- SaveMarkdownToStream[349].

See also:

- Saving and loading TRichView document[127].

6.1.1.3.118 TRichView.SaveMarkdownToStream

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the **Stream** as ***Markdown***.

```
function SaveMarkdownToStream(Stream: TStream;
  const Path: TRVUnicodeString[1099]; SelectionOnly: Boolean;
  CodePage: TRVCodePage[1062] = CP_UTF8): Boolean;
```

(introduced in version 19, changed in version 23)

Parameters

Path – path for saving images and other non-text items;

SelectionOnly – if *True*, only selected part of the document is saved.

CodePage – text encoding (UTF-8 by default)

Return value: "successful saving?"

See also properties:

- MarkdownProperties[245].

See also methods:

- SaveMarkdown[348].

See also:

- Saving and loading TRichView document[127].

6.1.1.3.119 TRichView.SavePicture

Saves a picture. This method is used when saving HTML, Markdown, and XML (using RichViewXML component) files.

```
function SavePicture(DocumentSaveFormat: TRVSaveFormat[1086];
  const ImagePrefix, Path: TRVUnicodeString[1099]; var ImageSaveNo: Integer;
  OverrideImages, InlineImage: Boolean; BackgroundColor: TRVColor[1063];
  gr: TRVGraphic[1032]; IsBackgroundImage = False): TRVUnicodeString[1099]; virtual;
```

(changed in version 18, 19, 21)

Parameters:

gr – the image to save.

DocumentSaveFormat – output document format, see TRVSaveFormat[1086] for possible values.

Path – path where to save the image (the last character must be '\').

If **DocumentSaveFormat**=*rvsfHTML* or *rvsfMarkdown,* 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[1138].

if **InlineImage** = *False:*

ImagesPrefix – the first part of an image file name. An image file name has format **ImagesPrefix** + <generated number>.

ImageSaveNo – default value of <generated number> (see above). If **OverrideImages** = *True*, **ImageSaveNo** is used. If **OverrideImages** = *False*, **ImageSaveNo** is incremented until the file with this name does not exist.

if **InlineImage** = *True:*

The image is not saved in a file, but its base64 representation is returned. It may be used to save the image directly in HTML.

BackgroundColor – color that is used for background (when converting transparent pictures to Jpeg or bitmap).

IsBackgroundImage should be *True* if the saved image is used as a background image.

You can use this method in the following events:

- OnHTMLSaveImage [383],
- OnSaveImage2 [410],
- OnSaveComponentToFile [405],
- OnSaveItemToFile [411] (SaveFormat=*rvsfHTML* or *rvsfMarkdown*).

If you use this method while saving HTML, it recommended to specify **ImagesPrefix** = ImagesPrefix parameter of the HTML saving functions, **OverrideImages** = *rvsoOverrideImages* in the Options parameter of the HTML saving functions.

If you use this method while saving Markdown, it recommended to specify **ImagesPrefix** = MarkdownProperties [245].ImagesPrefix [453], **OverrideImages** = *rvmdsoOverrideImages* in MarkdownProperties [245].SaveOptions [456].

See also:

- Export to HTML [132].

6.1.1.3.120 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[1099];
  SelectionOnly: Boolean):Boolean;
```

(introduced in version 1.3; changed in version 18)

Options for saving are in RTFOptions [252] property.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile [405] event.

By default, items tags [96] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink [418] event.

You can save additional information about non-text objects in OnWriteObjectProperties [420] event.

You can:

- completely change RTF code for certain items using OnSaveItemToFile [411] event;
- insert additional code in RTF in OnSaveRTFExtra [414] event;

- insert RTF codes [814] in text.

Return value: "successful saving?"

See also methods:
- SaveRTFToStream [351];
- CopyRTF [290];
- LoadRTF [334].

See also properties:
- RTFOptions [252];
- UseStyleTemplates [263].

See also events:
- OnWriteHyperlink [418];
- OnWriteObjectProperties [420];
- OnSaveComponentToFile [405];
- OnSaveRTFExtra [414];
- OnSaveItemToFile [411];
- OnProgress [392].

See also:
- Saving and loading in RichView [127].

6.1.1.3.121 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 [350], but saves to a stream instead of a file.

6.1.1.3.122 TRichView.SaveRVF

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the file **FileName** as RVF (RichView Format [129])

```
function SaveRVF(const FileName: TRVUnicodeString[1099];
  SelectionOnly: Boolean): Boolean;
```

(changed in version 18)

Setting for RVF saving can be changed in the TRichView component editor [224].

Return value: "successful saving?"

See also properties:
- RVFOptions [254].

See also methods:
- SaveRVFToStream [352];

- LoadRVF[336].

See also:

- Saving and loading RichView document[127];
- RVF[129].

6.1.1.3.123 TRichView.SaveRVFToStream

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the **Stream** as RVF (RichView Format[129]).

```
function SaveRVFToStream(Stream: TStream;
  SelectionOnly: Boolean): Boolean;
```

Setting for RVF saving can be changed in the TRichView component editor[224].

Return value: "successful saving?"

See also properties:

- RVFOptions[254].

See also methods:

- SaveRVF[351];
- LoadRVFFromStream[337];
- InsertRVFFromStream[323];
- AppendRVFFromStream[283].

See also:

- Saving and loading RichView document[127];
- RVF[129].

6.1.1.3.124 TRichView.SaveText -W

The methods export document to ANSI (**SaveText**) or Unicode (**SaveTextW**) text file.

```
function SaveText(const FileName: TRVUnicodeString[1099];
  LineWidth: Integer;  CodePage: Cardinal=CP_ACP): Boolean;
function SaveTextW(const FileName: TRVUnicodeString[1099];
  LineWidth: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName – output file name.

LineWidth is used for saving *breaks*[173] (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[260].DefCodePage[745] is used instead.

Return value: "successful saving?"

See also methods:

- SaveTextToStream, SaveTextToStreamW 353;
- LoadText, LoadTextW 338.

See also events:

- OnSaveImage2 410;
- OnSaveComponentToFile 405;
- OnSaveItemToFile 411.

See also:

- Saving and loading RichView documents 127.

6.1.1.3.125 TRichView.SaveTextToStream -W

The methods export document or selection as ANSI (**SaveTextToStream**) or Unicode (**SaveTextToStreamW**) text.

```
function SaveTextToStream(const Path: TRVUnicodeString 1099;
  Stream: TStream; LineWidth: Integer;
  SelectionOnly, TextOnly: Boolean;
  CodePage: Cardinal=CP_ACP):Boolean;
function SaveTextToStreamW(const Path: TRVUnicodeString 1099;
  Stream: TStream; LineWidth: Integer;
  SelectionOnly, TextOnly: Boolean):Boolean;
```

(introduced in version 1.3; changed in version 18)

Parameters

LineWidth is used for saving *breaks* 173 (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 410, OnSaveComponentToFile 405, and OnSaveItemToFile 411 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 260.DefCodePage 745 is used instead.

Return value: "successful saving?"

See also methods:

- SaveText, SaveTextW 352;
- LoadTextFromStream, LoadTextFromStreamW 339.

See also events:

- OnSaveImage2 410;

- OnSaveComponentToFile [405];
- OnSaveItemToFile [411].

See also:

- Saving and loading RichView documents [127].

6.1.1.3.126 TRichView.SearchText -A -W

The methods search for the substring **s** in the document.

```
function SearchText(s: String;
  SrchOptions: TRVSearchOptions[1091]): Boolean;
function SearchTextA(s: TRVAnsiString[1056];
  SrchOptions: TRVSearchOptions[1091]): Boolean;
function SearchTextW(s: TRVUnicodeString[1099];
  SrchOptions: TRVSearchOptions[1091]): Boolean;
```

When found, these methods select substring and scroll to it (this behavior can be modified in OnTextFound [418] event, see below).

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 *rvsroFromStored* is included in **SrchOptions**, the last stored search position [375] if used instead of selection.

If *rvsroMultiItem* is included, the search can match substrings of several text items [167]. 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** converts **s** to Unicode (using Style [260].DefCodePage [745]) and calls **SearchTextW**.

SearchText works:

- like **SearchTextA**, in Delphi 2007 and older
- like **SearchTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus [160]

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.

When the text is found, OnTextFound[418] event is called. If not disallowed in this event, the found text is selected. A selection requires a formatted document, so (unless you do your own processing in OnTextFound[418] and disallow a selection) these methods must be called only when the document is formatted.

Return value: Is substring found?

See also methods of TRichViewEdit:

- SearchText, -A, -W[553]

See also global functions:

- GetRVSearchOptions[1049].

See also:

- Searching in TRichView and TRichViewEdit[115].

6.1.1.3.127 TRichView.SelectAll

Selects the whole document for copying to the Clipboard

```
procedure SelectAll;
```

This method does not repaint document, call Invalidate.

In RichViewEdit[469] this method moves the caret to the end of document. The method generates OnSelect[416] event.

This method must be called only when the document is formatted.

See also methods:

- SetSelectionBounds[372];
- Deselect[294].

See also:

- Selecting in RichView document[112].

See also:

- Edit[1016] method of table cell.

6.1.1.3.128 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[174] containing the control **actrl**.

This method repaints document.

The method generates OnSelect[416] event.

In TRichViewEdit[469], if this control's item is resizable (see *rvepResizable* extra-property[1067]) and resizing with mouse is not forbidden (see *rvoNoImageResize* in EditorOptions[483]), 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[372];
- Deselect[294].

See also:

Selecting in RichView document[112].

6.1.1.3.129 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[416] 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[112].

6.1.1.3.130 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: TRVCoord[1064]);
```

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[235] property.

This method repaints document.

In RichViewEdit[469] this method moves the caret to the end of the selected word.

The method generates OnSelect[416] event.

Parameters:

(**X**,**Y**) – client coordinates (coordinates relative to the top left corner of TRichView 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 [247].

See also events:

- OnRVRightClick [404];
- OnRVDblClick [399].

See also properties:

- Delimiters [235].

See also methods:

- SelectAll [355];
- GetWordAt [322];
- SetSelectionBounds [372];
- GetSelText [321].

See also methods of TRichViewEdit:

- SelectCurrentWord [555].

See also:

- Selecting in RichView document [112].

6.1.1.3.131 TRichView.SetAddParagraphMode

Sets mode for adding items from new line for all Add*** methods [107].

```
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* [173] or tables [179]) 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 [143]

Example 1:

```
with MyRichView do
begin
  Clear [286];
```

```
  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);
  // error, do not add the first item in this mode!
  AddNL('First', 0, 0);
end;
```

Example 3:

```
with MyRichView do
begin
  Clear;
  SetAddParagraphMode(True); // default mode
  // ok, starting 1st paragraph with style 0
  AddNL('First',  0, 0);
  SetAddParagraphMode(False);
  // ok, adding this item from new line inside the 1st paragraph
  AddNL('Second', 0, 0);
  // error, do not add items with another paragraph style in this mode!
  AddNL('Third',  0, 1);
end;
```

See also

- Paragraphs [100];
- Building a document [107].

6.1.1.3.132 TRichView.SetBreakInfo

Changes main properties of the item of *break* [173] (horizontal line) type.

```
procedure SetBreakInfo(ItemNo: Integer; AWidth: TRVStyleLength[1094];
  AStyle: TRVBreakStyle[1059]; AColor: TRVColor[1063]; const ATag: TRVTag[1096]);
```

Parameters:

ItemNo – index of the item. The item must be of *break* [173] type (*rvsBreak* [1127]), otherwise the method raises ERichViewError [1014] exception. Items are indexed from 0 to ItemCount [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].SetBreakInfo.

AWidth – line width (or rectangle height). This value is measured in Style [260].Units [764].

AStyle – visual style of this *break*, see TRVBreakStyle [1059] for possible values.

AColor – line color. If it is equal to *rvclNone* [1105], Style [260].TextStyles [764][0].Color [810] is used.

ATag – *tag* of the item.You can use value returned by GetBreakInfo [296] or GetItemTag [310] for this item.

Instead of this method, you can use SetItemExtraIntPropertyEx[366] and SetItemTag[367] 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[366] and SetItemExtraStrProperty[367].

Example

```
RichView2.SetBreakInfo(ItemNo, 1, rvbsLine, clNone, 'Break Tag');
```

See also methods:

- GetBreakInfo[296];
- GetItemStyle[310];
- SetItemExtraIntProperty[366];
- SetItemExtraStrProperty[367].

See also properties:

- ItemCount[244].

See also methods of RichViewEdit:

- SetBreakInfoEd[557];
- SetCurrentBreakInfo[561].

See also:

- Modifying RichView items[117];
- Item types[90];
- *Tags*[96].

6.1.1.3.133 TRichView.SetBulletInfo

Changes main properties of the item of *bullet*[176] (image from ImageList) type.

VCL and LCL:

```
procedure SetBulletInfo(ItemNo: Integer; const AName: TRVUnicodeString[1099];
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure SetBulletInfo(ItemNo: Integer; const AName: TRVUnicodeString[1099];
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096];
  AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

Parameters:

ItemNo – index of the item. The item must be of *bullet*[176] or *hotspot*[178] type (*rvsBullet or rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are

not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].SetBulletInfo.

AName – name of *bullet*[176], any string without line break (CR, LF) characters. It can also be set using SetItemText[368] method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyEx[366] method.

ATag – *tag* of the item. You can use value returned by GetBulletInfo[297] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTag[367] method.

Additional parameters for FireMonkey version:

AImageWidth, AImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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[366] and SetItemExtraStrProperty[367].

Example

```
RichView1.SetBulletInfo(ItemNo, 'my image', 0, nil, 'bullet tag');
```

See also methods:

- GetBulletInfo[297];
- GetItemStyle[310];
- SetItemExtraIntProperty[366];
- SetItemExtraStrProperty[367].

See also properties:

- ItemCount[244].

See also methods of RichViewEdit:

- SetBulletInfoEd[558];
- SetCurrentBulletInfo[562].

See also:

- Modifying RichView items[117];
- Item types[90];
- *Tags*[96].

6.1.1.3.134 TRichView.SetCheckpointInfo

Creates a new *checkpoint* or modifies the existing *checkpoint* associated with the **ItemNo**-th item

```
procedure SetCheckpointInfo(ItemNo: Integer; const ATag: TRVTag[1096];
  const AName: TRVUnicodeString[1099]; ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].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[379] 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[247]).

See also methods:

- GetItemCheckpoint[305];
- RemoveCheckpoint[340].

See also properties:

- ItemCount[244].

See also methods of RichViewEdit:

- SetCheckpointInfoEd[560];
- SetCurrentCheckpointInfo[563].

See also:

- Modifying RichView items[117];
- *Checkpoints*[92];
- *Tags*[96].

6.1.1.3.135 TRichView.SetControlInfo

Changes main properties of the item of control[174] type.

```
function SetControlInfo(ItemNo: Integer; const AName: TRVUnicodeString[1099];
  AVAlign: TRVVAlign[1100]; const ATag: TRVTag[1096]): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control[174] type (*rvsComponent*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].SetControlInfo.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemText[368] method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign[1100] for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetControlInfo[301] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTag[367] 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[366] and SetItemExtraStrProperty[367].

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[301];
- GetItemStyle[310];
- Format[295];
- SetItemExtraIntProperty[366];
- SetItemExtraStrProperty[367].

See also properties:

- ItemCount[244].

See also methods of RichViewEdit:

- SetControlInfoEd[560];
- SetCurrentControlInfo[564].

See also:

- Modifying RichView items[117];
- Item types[90];
- *Tags*[96].

6.1.1.3.136 TRichView.SetFooter

Assigns the specified document for using as a footer.

```
procedure SetFooter(Footer: TCustomRVData[1014];
  HeaderType: TRVHFType[1078] = 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[715] 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[252];
- **Footer** will be saved after the main document in HTML, if HTMLSaveProperties[244].SaveHeaderAndFooter[445] = *True*.

Loading:

- when loading RVF, RTF, DocX containing a footer of this type, it will be read in **Footer**;

Each document assigned as a header/footer must be linked[260] to each own TRVStyle[739] component. Values of UseStyleTemplates[263] must be the same for the main document and all headers and footers. UseStyleTemplates[263] = *True*, MainRVStyle[756] of RVStyles of headers and footers must be equal to RVStyle of the main document (to allow them using the same collections of StyleTemplates[762]).

Example:

```
RichView1.Clear[286];
RichView2.Clear[286];
RichView1.RTFReadProperties.ReadDocParameters[465] := True;
RichView1.SetFooter(RichView2.RVData[254]);
RichView1.LoadRTF[334]('test.rtf');
RichView1.Format[295]; // contains document
RichView2.Format[295]; // contains header
RVPrint1.AssignSource[709](RichView1);
if RichView2.ItemCount[244]>0 then
  RVPrint1.SetFooter[715](RichView2.RVData)
else
  RVPrint1.SetFooter[715](nil);
RVPrint1.AssignDocParameters[708](RichView1.DocParameters[237]);
RVPrint1.FormatPages[711](rvdoAll);
RVPrint1.Print[713]('Doc with footer', 1, False);
```

See also:

- SetHeader[363].

See also properties of TRVStyle[739]:

- MainRVStyle[756].

6.1.1.3.137 TRichView.SetHeader

Assigns the specified document for using as a header.

```
procedure SetHeader(Header: TCustomRVData[1014];
  HeaderType: TRVHFType[1078] = 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[716] 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[252];
- **Header** will be saved before the main document in HTML, if HTMLSaveProperties[244] .SaveHeaderAndFooter[445] = *True*.

Loading:

- when loading RVF, DocX or RTF containing a header of this type, it will be read in **Header**;

Each document assigned as a header/footer must be linked[260] to each own TRVStyle[739] component. Values of UseStyleTemplates[263] must be the same for the main document and all headers and footers. UseStyleTemplates[263] = *True*, MainRVStyle[756] of RVStyles of headers and footers must be equal to RVStyle of the main document (to allow them using the same collections of StyleTemplates[762]).

Example:

```
RichView1.Clear[286];
RichView2.Clear[286];
RichView1.RTFReadProperties.ReadDocParameters[465] := True;
RichView1.SetHeader(RichView2.RVData[254]);
RichView1.LoadRTF[334]('test.rtf');
RichView1.Format[295]; // contains document
RichView2.Format[295]; // contains header
RVPrint1.AssignSource[709](RichView1);
if RichView2.ItemCount[244]>0 then
  RVPrint1.SetHeader[716](RichView2.RVData)
else
  RVPrint1.SetHeader[716](nil);
RVPrint1.AssignDocParameters[708](RichView1.DocParameters[237]);
RVPrint1.FormatPages[711](rvdoAll);
RVPrint1.Print[713]('Doc with header', 1, False);
```

See also:

- SetFooter[363].

See also properties of TRVStyle[739]:

- MainRVStyle[756].

6.1.1.3.138 TRichView.SetHotspotInfo

Changes main properties of the item of *hotspot*[178] (image from ImageList, hyperlink) type.

VCL and LCL:

```
procedure SetHotspotInfo(ItemNo: Integer;
```

```
  const AName: TRVUnicodeString[1099];
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList;
  const ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure SetHotspotInfo(ItemNo: Integer;
  const AName: TRVUnicodeString[1099];
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList;
  const ATag: TRVTag[1096];
  AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

Parameters:

ItemNo – index of the item. The item must be of *hotspot*[178] type (*rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].SetHotspotInfo.

AName – name of *hotspot*[178], any string without line break (CR, LF) characters. It can also be set using SetItemText[368] method.

AImageList – not used, reserved, set it to *nil*..

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyEx[366] 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[366] method.

ATag – *tag* of the item. You can use value returned by GetHotspotInfo[303] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTag[367] method.

AImageWidth, AImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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[366] and SetItemExtraStrProperty[367].

Example

```
RichView1.SetHotspotInfo(ItemNo, 'my image', 0, 0, nil, 'hotspot tag');
```

See also methods:

- GetHotspotInfo[303];
- GetItemStyle[310];
- SetItemExtraIntProperty[366];

- SetItemExtraStrProperty[367].

See also properties:

- ItemCount[244].

See also methods of RichViewEdit:

- SetHotspotInfoEd[570];
- SetCurrentHotspotInfo[565].

See also:

- Modifying RichView items[117];
- Item types[90];
- *Tags*[96].

6.1.1.3.139 TRichView.SetItemExtraIntProperty -Ex

The methods change a value of the specified integer property of the **ItemNo**-th item.

```
function SetItemExtraIntProperty(ItemNo: Integer;
  Prop: TRVExtraItemProperty[1067]; 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[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].SetItemExtraIntProperty[Ex]

In **SetItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[1067]. 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[1119]

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][307];
- SetItemExtra**Str**Property[367];
- Format[295].

See also methods of TRichViewEdit

- SetItemExtraIntProperty[Ex]Ed[572];
- SetCurrentItemExtraIntProperty[Ex][566].

6.1.1.3.140 TRichView.SetItemExtraStrProperty -Ex

The methods change a value of the specified additional string property of the **ItemNo**-th item.

```
function SetItemExtraStrProperty(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty[1072];
  const Value: TRVUnicodeString[1099]): Boolean;
function SetItemExtraStrProperty(ItemNo, Prop: Integer;
  const Value: TRVUnicodeString[1099]): 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[1072]. 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[1124]

ItemNo – index of the item, from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].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][308];
- SetItemExtra**Int**Property[Ex][366].

See also methods of TRichViewEdit:

- SetItemExtraStrProperty[Ex]Ed[573];
- SetCurrentItemExtraStrProperty[Ex][567].

6.1.1.3.141 TRichView.SetItemTag

Changing *tag* of the **ItemNo**-th item

```
procedure SetItemTag(ItemNo: Integer; const ATag: TRVTag[1096]);
```

Items are indexed from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[964].SetItemTag.

Method type: viewer-style. Reformatting or repainting is not necessary.

Example

```
RichView1.SetItemTag(ItemNo, 'tag #1');
```

See also methods:

- GetItemTag [310].

See also properties:

- ItemCount [244].

See also methods of RichViewEdit:

- SetItemTagEd [574];
- SetCurrentTag [570].

See also:

- Modifying RichView items [117];
- *Tags* [96].

6.1.1.3.142 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[1056]);
procedure SetItemTextW(ItemNo: Integer; const s: TRVUnicodeString[1099]);
```

(Introduced in version 1.3, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].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** converts ANSI to Unicode. For a text item, a code page for conversion is calculated basing on the its Charset [809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style [260].DefCodePage [745] is used.

SetItemText works:

- like **SetItemTextA**, in Delphi 2007 and older
- like **SetItemTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus [160]

Methods type: viewer-style. Reformatting and repainting are required only for text items.

See also methods:

- GetItemText -W -A [310];
- Format [295].

See also methods of TCustomRichViewEdit:

- SetItemTextEd -A -W [574];
- SetCurrentItemText -A -W [568].

See also:

- Modifying RichView Items [117];
- Unicode in RichView [135].

6.1.1.3.143 TRichView.SetItemVAlign

Changing vertical alignment (position relative to the line) of the **ItemNo**-th item

```
procedure SetItemVAlign(ItemNo: Integer; VAlign: TRVVAlign[1100]);
```

(introduced in version 12)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount [244]-1. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].SetItemVAlign.

VAlign – new value of vertical alignment.

This method must not be called for text items [167], tables [179], *breaks* [173], tabs [168], list markers [180], footnotes [185] and endnotes [184].

Method type: viewer-style. Requires reformatting and repainting.

See also methods:

- GetItemVAlign [311].

See also properties:

- ItemCount [244].

See also methods of RichViewEdit:

- SetItemVAlignEd [575];
- SetCurrentItemVAlign [569].

See also:

- Modifying RichView items [117]

6.1.1.3.144 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 [244]-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 [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].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[244], 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[143]).

AListNo – style of this list maker. This is an index in the collection of list styles (Style[260].ListStyles[755]).

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[**AListNo**].Levels[842]).

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[260].ParaStyles[757]). 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[277] to add empty text item after the list marker. See also: rules for valid documents[143]

Return value: item index of the marker.

See also methods:

- GetListMarkerInfo[315];
- Format[295].

See also properties:

- Style[260];
- ItemCount[244].

See also methods of TCustomRichViewEdit

- ApplyListStyle[498];
- RemoveLists[552];
- ChangeListLevels[508].

See also:

- List Markers[180].

6.1.1.3.145 TRichView.SetPictureInfo

Changes main properties of the item of picture[169] or *hot-picture*[172] type.

```
function SetPictureInfo(ItemNo: Integer; const  AName: TRVUnicodeString[1099];
  Agr: TRVGraphic[1032]; AVAlign: TRVVAlign[1100];
  const ATag: TRVTag[1096]): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture [169] or *hot-picture* [172] type (*rvsPicture or rvsHotPicture* [1127]), otherwise the method raises ERichViewError [1014] exception. Items are indexed from 0 to ItemCount [244]-1, GetItemStyle [310] returns type of item. Items of subdocuments (table cells [951]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData [964].SetPictureInfo.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemText [368] 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 [318] for this item (the image is shared if *rvepShared* extra item property [1067] is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetPictureInfo [318] 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 [318] or GetItemTag [310] for this item. The *tag* can also be set by SetItemTag [367] 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 [366] and SetItemExtraStrProperty [367] (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, '');
// incorrect! you cannot assign
// the same graphic object to different items
RichView1.SetPictureInfo(ItemNo2, '', bmp, rvvaBaseline, '');
```

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(366)(ItemNo1, rvepShared, 1);
RichView1.SetPictureInfo(ItemNo2, '', bmp, rvvaBaseline, '');
RichView1.SetItemExtraIntProperty(366)(ItemNo2, rvepShared, 1);
// you must free bmp yourself when RichView1 is cleared
```

See also methods:

- GetPictureInfo(318);
- GetItemStyle(310);
- Format(295);
- SetItemExtraIntProperty(366);
- SetItemExtraStrProperty(367).

See also properties:

- ItemCount(244).

See also methods of RichViewEdit:

- SetPictureInfoEd(576);
- SetCurrentPictureInfo(569).

See also:

- Modifying RichView items(117);
- Item types(90);
- *Tags*(96).

6.1.1.3.146 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(167), 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(810)>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(810)>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[469], the caret is always at the end of the selection, so this method can be used to move caret.

Selection in table[179] cells

Before making selecting inside a table cell, activates its editing:

```
var RVData: TCustomRVFormattedData[1014];

RVData := TCustomRVFormattedData[1014](Table.Cells[912][r,c].Edit[1016]);
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[416] event.

There are two useful methods to work with SetSelectionBounds: GetOffsBeforeItem[317] and GetOffsAfterItem[316].

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[1047] unit instead.

See also methods:

- SelectAll[355];
- Deselect[294];
- GetSelectionBounds[320].

See also events:

- OnSelect[416].

See also:

- Working with selection[112];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[1133];
- Example how to move caret to the beginning of paragraph[602].

See also:

- Edit[1016] method of table cell.

6.1.1.3.147 TRichView.StartAnimation

Starts animation of pictures.

```
procedure StartAnimation;
```

(introduced in version 10)

Alternatively, animation may start automatically when document is formatted[295], if AnimationMode[228] = *rvaniOnFormat*.

See also methods:

- ResetAnimation[341];
- StopAnimation[375].

See also properties:

- AnimationMode[228].

See also:

- Animation in TRichView[124];
- Pictures[169];
- *Hot-Pictures*[172].

6.1.1.3.148 TRichView.StartLiveSpelling

Starts live spelling check (runs a live spelling thread).

```
procedure StartLiveSpelling;
```

(introduced in v1.9)

A check starts only if:

- OnSpellingCheck[416] event is assigned, or
- [in FireMonkey, in TRichViewEdit[469]] CheckSpelling[480] = *True*, and the current platform provides a spelling checking service.

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[487].

See also:

- Live spelling check in TRichView[138]
- TRVSpellChecker[728].StartLiveSpelling

6.1.1.3.149 TRichView.StopAnimation

Stops animation of pictures.

```
procedure StopAnimation;
```

(introduced in version 10)

Clear[286] may stop animations as well.

See also methods:

- StartAnimation[374];
- ResetAnimation[341].

See also properties:

- AnimationMode[228].

See also:

- Animation in TRichView[124];
- Pictures[169];
- *Hot-Pictures*[172].

6.1.1.3.150 TRichView.StoreSearchResult

Stores the position in the document to allow searching from this place.

```
procedure StoreSearchResult(ARVData: TCustomRVData[1014];
  StartItemNo, StartItemOffs, EndItemNo, EndItemOffs: Integer;
  Invert: Boolean);
```

(introduced in version 21)

This method is useful only if you want to search without selecting found strings.
You can call this method from OnTextFound[418] event to store the position of the search result. Then you can continue searching from this position by calling TRichView.SearchText[354] with *rvsoFromStored*[1091] in the Options parameter (or TRichViewEdit.SearchText[553] with *rvseoFromStored*[1066] in the Options parameter

See also:

- Searching in TRichView and TRichViewEdit[115].

6.1.1.3.151 TRichView.UpdatePaletteInfo

Call this method when screen color resolution changed (see WM_DISPLAYCHANGE window message). You need not to call this method if DoInPaletteMode[241]=rvpaDoNothing.

```
procedure UpdatePaletteInfo;
```

Example[242]

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[241].

See also methods of TRVPrint:

- UpdatePaletteInfo [884].

6.1.1.4 Events

Derived from TCustomRichView

- OnAddStyle [377]
- OnAfterDrawImage [378]
- OnAssignImageFileName [379]
- OnCheckpointVisible [379]
- OnCMHintShow [380] [VCL,LCL]
- OnControlAction [380]
- OnCopy [381]
- OnGetItemCursor [382]
  OnGetSpellingSuggestions [382] [FMX]
- OnHTMLSaveImage [383]
- OnImportFile [385]
- OnImportPicture [386]
- OnItemAction [387]
- OnItemHint [388]
- OnJump [390]
- OnLoadCustomFormat [390]
- OnLoadDocument [391]
- OnNewDocument [391]
- OnPaint [392]
- OnProgress [392]
- OnReadField [393]
- OnReadHyperlink [395]
- OnReadMergeField [398]
- OnRVDblClick [399]
- OnRVFControlNeeded [400]
- OnRVFImageListNeeded [400]
- OnRVFPictureNeeded [401]
- OnRVMouseDown [402]
- OnRVMouseMove [403]
- OnRVMouseUp [403]
- OnRVRightClick [404]
- OnSaveComponentToFile [405]
- OnSaveDocXExtra [407]
- OnSaveHTMLExtra [409]
- OnSaveImage2 [410]
- OnSaveItemToFile [411]
  OnSaveParaToHTML [413]
- OnSaveRTFExtra [414]
- OnSelect [416]
- OnSpellingCheck [416]
- OnStyleTemplatesChange [417]

- OnTextFound 418
- OnWriteHyperlink 418
- OnWriteObjectProperties 420

Derived from TRVScroller 869

- OnHScrolled 877
- OnVScrolled 877

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- OnApplyStyleLookup [FMX]
- OnClick
- OnContextPopup [VCL,LCL]
- OnDragDrop
- OnDragEnd [FMX]
- OnDragEnter [FMX]
- OnDragLeave [FMX]
- OnDragOver
- OnEndDrag [VCL,LCL]
- OnEnter
- OnExit
- OnGesture (D2010+) [VCL,FMX]
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseEnter [FMX]
- OnMouseLeave [FMX]
- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown [VCL,LCL]
- OnMouseWheelUp [VCL,LCL]
- OnResize
- OnResized [FMX]
- OnStartDrag [VCL,LCL]

6.1.1.4.1 TRichView.OnAddStyle

Occurs when new style is added.

```
type
  TRVAddStyleEvent = procedure (Sender: TCustomRichView 216;
    StyleInfo: TCustomRVInfo 803) of object;

property OnAddStyle: TRVAddStyleEvent;
```

(introduced in version 11)

This event occurs when a new style is added (to TextStyles 764, ParaStyles 757, or ListStyles 755 collections of the linked TRVStyle 739 component) as a result of:

- RTF (Rich Text Format) reading (both loading and insertion)

- Markdown reading (both loading and insertion)
- RVF (RichView Format) insertion;
- (if style templates are used[263]) auto-correction of ParaStyleTemplateId[815] property of text styles;
- (in editor) changing the current text style on keyboard language change (see *rvoAutoSwitchLang* in EditorOptions[483]);
- (in editor) ApplyStyleTemplate[501], ApplyParaStyleTemplate[500], ApplyTextStyleTemplate[502];
- (in editor, if style templates are used[263]) when a user presses **Enter** (because of applying NextId[846] 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.

6.1.1.4.2 TRichView.OnAfterDrawImage

Allows custom drawing on pictures[169] and hot-pictures[172].

VCL and LCL:

```
type
  TRVAfterDrawImageEvent = procedure (Sender: TCustomRichView[216];
    ARVData: TCustomRVData[1014];
    AItem: TCustomRVItemInfo; AState: TRVItemDrawStates;
    const ARect: TRVCoordRect[1064]; ACanvas: TCanvas) of object;
```

FireMonkey:

```
type
  TRVAfterDrawImageEvent = procedure (Sender: TCustomRichView[216];
    ARVData: TCustomRVData[1014];
    AItem: TCustomRVItemInfo; AState: TRVItemDrawStates;
    const ARect: TRVCoordRect[1064]; ACanvas: TRVFMXCanvas[1018]) of object;
```

```
property OnAfterDrawImage: TRVAfterDrawImageEvent;
```

(introduced in version 20)

Parameters:

ACanvas – canvas for painting.

ARVData – a document containing the item.

AItem – an item that is being drawn.

ARect – image rectangle on **ACanvas**; it does not include spacing and borders.

This event occurs when the image is already drawn. It allows drawing additional content on top of it.

See also:

- TCustomRVPrint.OnAfterPrintImage[885]

6.1.1.4.3 TRichView.OnAssignImageFileName

Occurs when inserting an image file in TRichView.

```
type
  TRVAssignImageFileNameEvent = procedure (Sender: TCustomRichView[216];
    var FileName: TRVUnicodeString[1099]) of object;

property OnAssignImageFileName: TRVAssignImageFileNameEvent;
```

(introduced in version 16; changed in version 18)

This event occurs if *rvoAssignImageFileNames* is included in Options[247].

In this event, you can modify **FileName** (for example, change a full path to a relative path).

This event may happen on:

- RTF loading[334] (reading external images),
- DocX loading[326] (reading external images),
- Markdown loading[332] (reading external images),
- HTML loading (using importer components),
- pasting files[543], accepting files and links to images as a result of drag&drop[123],
- RichViewActions (actions for inserting images and assigning background images of tables).

The following properties are set:

- table.BackgroundImageFileName[905]
- cell.BackgroundImageFileName[953]
- rvespImageFileName[1072] extra item properties of pictures[169] and hot-pictures[172].

6.1.1.4.4 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[233] = *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[233].

See also methods extracting information from CheckpointData:

- GetCheckpointInfo[299];

- GetCheckpointXY[300];
- GetCheckpointYEx[301];
- GetCheckpointItemNo[299];
- GetCheckpointNo[299].

See also

- Checkpoints[92].

6.1.1.4.5 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[388]

6.1.1.4.6 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 [260] .RVData (must be typecasted to TCustomRVData [1014]) and **ItemNo** parameter. **ItemNo** is an index of item (containing **ctrl**) in the document **Sender**.Style.RVData (it may be RVData [254] of the main RichView (**Sender**), cell [951], or RVData [254] 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 [387].

6.1.1.4.7 TRichView.OnCopy

Occurs when copying to the Clipboard

```
property OnCopy: TNotifyEvent;
```

(Introduced in version 1.4)

Occurs inside CopyDef [288] and Copy [288] 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 [594];
- Working with Clipboard [113].

6.1.1.4.8 TRichView.OnGetItemCursor

Allows defining custom custom cursors for items.

```
type
  TRVGetItemCursorEvent = procedure (Sender: TCustomRichView[216];
    RVData: TCustomRVData[1014]; 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[254], or table cell, or cell inplace editor's RVData[254])

ItemNo – index of the item inside **RVData**.

Output Parameter

Cursor – cursor to display.

See also:
- OnItemHint[388].

6.1.1.4.9 TRichView.OnGetSpellingSuggestions

This event requests a list of suggestions for the misspelled word.

```
type
  TRVGetSpellingSuggestionsEvent = procedure(
    Sender: TCustomRichView[216];
    const AWord: TRVUnicodeString[1099]; StyleNo: Integer;
    out ASuggestions: TArray<String>) of object;

property OnGetSpellingSuggestions: TRVGetSpellingSuggestionsEvent;
```

(introduced in v21)

This event is used when the component wants to display a popup menu and to populate it with items that offer corrections for the misspelled word.

In the current version, this event is not used in TRichView, only in TRichViewEdit[469].

If a platform spelling check service is used (see CheckSpelling[480] property), this event is not necessary: suggestions are received from a platform service. However, if this event is assigned, it is used instead of a platform service.

If PopupMenu property is not assigned, items are added to the built-in default popup menu.

If PopupMenu property is assigned, you can use AddSpellingMenuItems[497] to add items in any popup menu.

Input parameters:

AWord is a word to correct.

StyleNo is a style of text item[167] containing this word (index in the collection **Sender**.Style[260].TextStyles[764]).

Output parameter:

ASuggestions, a list of suggestions to replace **AWord** in the document.

See also:

- Live spelling check in TRichView[138];
- OnSpellingCheck[416] event

6.1.1.4.10 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[1099];
      BackgroundColor: TRVColor[1063]; 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[169], *hot-pictures*[172],
- *bullets*[176], *hotspots*[178],
- list markers[180] (paragraph bullets) with images,
- tables[179] (saving background image[904]),
- for background image[229] and cells' background images[953].

There is more easy-to-use event: OnSaveImage2[410].

By default:

Images are saved in separate files. Png, Gif, Jpeg images are saved in the corresponding format. Images of non-web formats are saved as Jpeg images. If you do not want to convert some additional image class to Jpeg, you can register it as HTML graphic format usiing RVGraphicHandler[1033]. File names of saved images are composed as ImagesPrefix + Number + '.' + FileExtension, where

- ImagesPrefix – HTMLSaveProperties[244].ImagesPrefix[443];
- Number – some number making this file name unique (image files can override existing ones or not, depending on *rvhtmlsioOverrideImages in* HTMLSaveProperties[244].ImageOptions[442]).

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[233]

/TRVStyle.Color[744]), color of cell (TRVTableCellData.Color[957]) or color of paragraph background (TRVBackroundRect.Color[852]).

You can define formats that must not be converted to Jpeg[1138].

Input Parameters

RVData – document containing item that is being saved to image; it can be **Sender**.RVData[254], table cell[951], 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[346]).

BackgroundColor – color of background below the item (can be ignored, if you can preserve transparency of image)

Location – if *rvhtmlsioUseItemImageFileNames* is included in HTMLSaveProperties[244].ImageOptions[442], this is a path to the image defined in the *rvespImageFileName* item property[1072], 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 *rvhtmlsioUseItemImageFileNames* is included in HTMLSaveProperties[244].ImageOptions[442] and value of *rvespImageFileName* item property[1072] is not empty, this value is written in HTML (relative to **Path**); otherwise
- OnSaveImage2[410] 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[1099];
  var imgSaveNo: Integer; OverrideFiles: Boolean): TRVUnicodeString[1099];
```

where

ImagesPrefix – the first part of image file name (for example, you can set it to HTMLSaveProperties[244].ImagesPrefix[443]);

Path – path for the image (for example, you can use **Path** parameter of this event);

Ext – file extension, for example '.jpg' (you can use RVGraphicHandler[1125].GetGraphicExt() or GetGraphicExtByType());

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 [349].

See also events:

- OnSaveImage2 [410];
- OnSaveItemToFile [411].

See also

- Export to HTML [132].

6.1.1.4.11 TRichView.OnImportFile

Occurs when loading a file or a stream containing links to other files.

```
type
  TRVImportFileEvent = procedure(
    Sender: TCustomRichView;
    const Location: TRVUnicodeString[1099];
    out Content: TStream) of object;

property OnImportFile: TRVImportFileEvent;
```

(introduced in version 21)

This event may occur when loading HTML that has external style sheets.

If these links are local (like 'c:\images\styles.css'), TRichView can load these files itself. But if files are on the Internet (like 'https://www.trichview.com/styles.css'), it makes sense to process this event and to download the image.

Input parameters

Location – path to the file (local or URL);

Output parameters

Content – a stream that contains the file content. If you leave it equal to *nil*, TRichView will attempt to load the file itself. If you provided this stream when processing this event, do not free it (it will be freed by TRichView).

If you use RichViewActions, they can use downloader components to download files 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 [580] events.

6.1.1.4.12 TRichView.OnImportPicture

Occurs when loading a file or a stream containing links to image files.

```
type
  TRVImportPictureEvent =
    procedure (Sender: TCustomRichView;
    const Location: TRVUnicodeString[1099]; Width, Height: Integer;
    var Graphic: TRVGraphic[1032]) of object;

property OnImportPicture: TRVImportPictureEvent;
```

(introduced in version 1.8; changed in version 18)

This event may occur:

- when loading RTF or DocX. Usually pictures are saved inside RTF/DocX file/stream. But sometimes RTF/DocX contains links to external image files (file names or URLs).
- when loading HTML or Markdown
- when loading XML (using TRichViewXML component)
- when accepting URL to an image as a result of drag&drop, and *rvoDragDropPicturesFromLinks* is include in EditorOptions[483].

If these links are local (like 'c:\images\image.bmp'), TRichView can load these files itself. But if images are on the Internet (like 'https://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.

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[580] events.

Example: downloading images from http location using Indy (IdHTTP1: TIdHTTP)

```
uses RVFuncs;

{ TMyRichView.OnImportPicture }
procedure TMyForm.MyRichViewImportPicture(Sender: TCustomRichView;
  const Location: TRVUnicodeString[1099]; Width, Height: Integer;
  var Graphic: TRVGraphic[1032]);
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[1125].LoadFromStream(Stream);
    except
      Graphic := nil;
    end;
    Stream.Free;
  end;
end;
```

6.1.1.4.13 TRichView.OnItemAction

Occurs on insertion, deletion and some other modifications of item

```
type
  TRVItemAction = (rviaInserting, rviaInserted,
    rviaTextModifying, rviaDestroying, rviaMovingToUndoList)

  TRVItemActionEvent = procedure (Sender: TCustomRichView[216];
    ItemAction: TRVItemAction; Item: TCustomRVItemInfo[899];
    var Text: TRVUnicodeString[1099]; RVData: TCustomRVData[1014]) of object;

property OnItemAction: TRVItemActionEvent;
```

(introduced in version 1.7; modified in v18)

Parameters

Item – item[899] 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[254], table cell[951], 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.

ItemAction	Meaning
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[1099]; 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[380].

See also methods:

- GetItem[304];
- GetItemNo[309].

6.1.1.4.14 TRichView.OnItemHint

Allows to define custom popup hints for items.

```
type
  TRVItemHintEvent = procedure (Sender: TCustomRichView[216];
    RVData: TCustomRVData[1014]; ItemNo: Integer;
    var HintText: TRVUnicodeString[1099]) of object;

property OnItemHint: TRVItemHintEvent;
```

This event occurs when mouse pointer moves over the item.

Parameters

RVData – document containing this item (RichView.RVData[254], or table cell, or cell inplace editor's RVData[254])

ItemNo – index of the item inside **RVData**.

HintText – on input, it's a value of *rvespHint* item property[1072]. On output, it's a message that will be displayed.

Item popup hint can be displayed if *rvoShowItemHints* is in RichView.Options[247] and RichView.ShowHint=*True*.

Cell Hint[958]s cannot be altered in this event.

See also:

- OnGetItemCursor[382].

Example [VCL and LCL]

Besides the default hint, this code displays:

- for hyperlinks: tag;
- for images: Delphi class type, width and height.

```
procedure TMyForm.MyRichViewEditItemHint(Sender: TCustomRichView[216];
  RVData: TCustomRVData[1014]; ItemNo: Integer;
  var HintText: TRVUnicodeString[1099]);
var Tag: TRVTag[1096];
    VAlign: TRVVAlign[1100];
    Name: TRVUnicodeString[1099];
    gr: TGraphic;
begin
  if RVData.GetItem[304](ItemNo).GetBoolValueEx(
    rvbpJump, Sender.Style[260]) then
  begin
    if HintText<>'' then
      HintText := HintText+#13;
    HintText := HintText + 'Target: '+RVData.GetItemTag[310](ItemNo);
  end;
  if (RVData.GetItemStyle[310](ItemNo)=rvsPicture[1127]) or
     (RVData.GetItemStyle[310](ItemNo)=rvsHotPicture[1127]) then
  begin
    if HintText<>'' then
      HintText := HintText+#13;
    RVData.GetPictureInfo[318](ItemNo, Name, gr, VAlign, Tag);
    HintText := HintText+gr.ClassName+
      ' ('+IntToStr(gr.Width)+','+IntToStr(gr.Height)+')';
  end;
end;
```

6.1.1.4.15 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[243], the next link has **id**=FirstJumpNo[243]+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[483]) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly[488]=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[262]<>*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[1014];
    s: String;
begin
  MyRichView.GetJumpPointLocation[312](id, RVData, ItemNo);
  s := RVData.GetItemTag[310](ItemNo);
  Application.MessageBox(PChar(s), 'Link', MB_OK or MB_ICONINFORMATION);
end;
```

See also events:

- OnRVMouseMove[403].

See also:

- RichView hypertext[97].

6.1.1.4.16 TRichView.OnLoadCustomFormat

This event allows loading data from a database field in your own format.

```
type
  TRVCustomFormatEvent = procedure (Sender: TCustomRichView[216];
    Stream: TStream; var DoDefault: Boolean) of object;

property OnLoadCustomFormat: TRVCustomFormatEvent;
```

(introduced in version 10)

This event occurs when loading a document from a database field:

- linked using LiveBindings (Delphi XE2+, see Document[238] property)

- by TDBRichView[604] and TDBRichViewEdit[616] components

Input parameters:

Stream contains data from a database field. 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:

TRichViewEdit[469].OnSaveCustomFormat[595]

6.1.1.4.17 TRichView.OnLoadDocument

Occurs after loading a document from a database field.

```
property OnLoadDocument: TNotifyEvent;
```

(introduced in v1.9)

This event occurs when loading a document from a database field:

- linked using LiveBindings (Delphi XE2+, see Document[238] property)
- by TDBRichView[604] and TDBRichViewEdit[616] components

This event allows to preprocess the loaded document before displaying.

See also:

- OnNewDocument[391].

6.1.1.4.18 TRichView.OnNewDocument

Occurs when creating or before loading a document.

```
property OnNewDocument: TNotifyEvent;
```

(introduced in v1.9)

This event occurs when loading a document from a database field:

- linked using LiveBindings (Delphi XE2+, see Document[238] property)
- by TDBRichView[604] and TDBRichViewEdit[616] components

It 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[391].

6.1.1.4.19 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 [719], OnPagePostpaint [718]).

VCL/LCL: use client coordinates to draw. Point(0,0) is a the top left corner of the TRichView window.

FireMonkey: use document coordinates to draw. PointF(HScrollPos [872], VScrollPos [874]) is a the top left corner of the TRichView window.

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.1.1.4.20 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,
    rvloMarkdownRead, rvloMarkdownWrite,
    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, v19)

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[129];
- reading/writing HTML;
- writing DocX;
- reading/writing a plain ANSI or Unicode text;
- when calling TRVOfficeConverter.ImportRV[694] and ExportRV[693].

Note: the event is called only if the document/source file is large enough.

See also:

- TRVOfficeConverter.OnConverting[695]

6.1.1.4.21 TRichView.OnReadField

Occurs when reading fields from RTF or DocX file.

```
type // defined in RVStyle unit

  TRVRTFFieldResultAction =
    (rvrtffraInsert, rvrtffraCollectText);

  TRVFieldResultData = record
    FieldResultAction: TRVRTFFieldResultAction;
    Item: TObject;
    Text: TRVUnicodeString[1099];
    Tag: TRVTag;
    StyleNo: Integer;
  end;

  TRVFieldContext = record
    FirstTime: Boolean;
    ParaNo: Integer;
    FieldResultText: TRVUnicodeString[1099];
  end;

type
  TRVReadFieldEvent = procedure (
    Sender: TCustomRichView[216];
    const FieldCode: TRVUnicodeString[1099];
    Context: TRVFieldContext;
    DocFormat: TRVLoadFormat;
    out FieldResult: TRVFieldResultData;
    var DoDefault: Boolean) of object;

property OnReadField: TRVReadFieldEvent;
```

(introduced in version 22)

This event occurs when reading fields from RTF or DocX files. It occurs only for fields that are not processed by TRichView itself.

Input parameters:

FieldCode – field code, including RTF/DocX field name and parameters

DocFormat – format of loaded document, *rvlfRTF* or *rvlfDocX*

Context – additional information describing the current state of RTF/DocX reading

DoDefault = *True*

Output parameters:

DoDefault – if *False*, TRichView should perform the action described in **FieldResult**; if *True*, TRichView should perform default processing (i.e., reading the field result from RTF/DocX, and inserting it as normal content).

FieldResult describes the action that must be performed by TRichView in the place of this field; used only if **DoDefault** = *False*.

Option 1: inserting an item

If you assign **FieldResult**.**FieldResultAction** = *rvrtffraInsert*, the content described in **FieldResult** must be inserted in TRichView.

If **FieldResult.Item** <> *nil*, it must be an object of class inherited from TCustomRVItemInfo[899]. This item will be inserted in TRichView document, **FieldResult**.**Text** will be used as an item text. If this item has an associated text style (for example, TRVLabelItemInfo[970] has TextStyleNo[973] property), you can assign *rvsDefStyle*[1127] to the index of text style; TRichView will replace this value to the index representing the current text attributes in RTF/DocX.

Option 2: inserting text

If you assign **FieldResult**.**FieldResultAction** = *rvrtffraInsert*, the content described in FieldResult must be inserted in TRichView.

If **FieldResult.Item** = *nil*, text **FieldResult.Text** will be inserted. If **FieldResult.StyleNo** >=0, it defines a text style of this text (index in Style[260].TextStyles[764] collection). If **FieldResult.StyleNo** < 0 (or is equal to *rvsDefStyle*[1127]), the current RTF/DocX text attributes are used. **FieldResult.Text** may be multiline, but cannot contain tab (#9) characters. If **FieldResult.Tag** <> '', this tag[96] is assigned to this text.

Option 3: collecting text

If you assign **FieldResult**.**FieldResultAction** = *rvrtffraCollectText*, nothing will be inserted after calling this event. Instead, the component will read the field result, collecting text from it. The event will be called for the second time for the same field, with **Context.FirstTime** = *False*, and the collected text in **Context.FieldResultText**.

Additional information in Context

Fields of **Context** parameter:

FirstTime = *True* if the event is called for the first time for the given field. *False* if it is called after collecting text from the field result.

FieldResultText – collected text, valid only if **FirstTime** = *False*.

ParaNo – index of paragraph style (in Style[260].ParaStyles[757] collection) representing the current paragraph attributes in RTF/DocX file. While paragraph index will be assigned to the inserted content by TRichView, this property can be useful if you assign your own text style (to **FieldResult.StyleNo**; or a text style associated with **FieldResult.Item**). Knowing ParaNo, if StyleTemplates are used,[263] you can assign a text style with the proper value of ParaStyleTemplateId[815].

6.1.1.4.22 TRichView.OnReadHyperlink

Occurs when reading hyperlinks (for example, from RTF file)

```
type
  TRVReadHyperlink =
    procedure (Sender: TCustomRichView;
      const Target, Extras: TRVUnicodeString[1099];
      DocFormat: TRVLoadFormat[1081];
     var StyleNo: Integer; var ItemTag: TRVTag[1096];
     var ItemName: TRVUnicodeString[1099]) of object;

property OnReadHyperlink: TRVReadHyperlink;
```

(changed in version 18)

By default, hyperlink targets is saved in the item tag[96]. 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[1072], and adds '#' to the beginning of links to bookmarks.

DocFormat – format of loaded document.

DocFormat	Meaning
rvlfRTF	The event occurs when reading RTF files (or streams) containing hyperlinks (RTF loading, inserting, pasting, or accepting on drag&drop).

DocFormat	Meaning
rvlfDocX	The event occurs when reading DocX (Microsoft Word Document) files (or streams) containing hyperlinks (DocX loading or inserting).
rvlfURL	The event occurs: 1) on drag&drop, when accepting URL dropped to TRichViewEdit[469] (*rvddURL* must be in AcceptDragDropFormats[478]). 2) when pasting from the Clipboard, when pasting URL to TRichViewEdit[469] (*rvddURL* must be in AccepPasteFormats[478]).
rvlfHTML	The event occurs when reading HTML files (or streams) containing hyperlinks (HTML loading, inserting or pasting).
rvlfMarkdown	The event occurs when reading Markdown files (or streams) containing hyperlinks (Markdown loading, inserting or pasting).

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[764] of this text. You can modify it to point to another (hypertext[823]) 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[96] 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[96].

```
procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[1099]; DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag[1096];
```

```
  var ItemName: TRVUnicodeString[1099]);
begin
  ItemTag := Target;
end;
```

More complicated example

This example assumes that you included *rvddURL* in AcceptDragDropFormat[478] and want to accept hyperlinks. Besides, it's assumed that "Allow adding styles dynamically" is set in the component editor[224] (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[1099]; DocFormat: TRVLoadFormat[1081];
  var StyleNo: Integer; var ItemTag: TRVTag[1096];
  var ItemName: TRVUnicodeString[1099]);
var FontStyle: TFontInfo[821];
begin
  ItemTag := Target;
  if DocFormat=rvddURL then
  begin
    FontStyle := TFontInfo[821].Create(nil);
    FontStyle.Assign(Sender.Style[260].TextStyles[764][StyleNo]);
    FontStyle.Color[810] := clBlue;
    FontStyle.Style[817] := FontStyle.Style + [fsUnderline];
    FontStyle.Jump[823] := True;
    StyleNo := Sender.Style.FindTextStyle[771](FontStyle);
    FontStyle.Free;
  end;
end;
```

The same example using RichViewActions:

```
procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[1099]; DocFormat: TRVLoadFormat[1081];
  var StyleNo: Integer; var ItemTag: TRVTag[1096];
  var ItemName: TRVUnicodeString[1099]);
begin
  ItemTag := Target;
  if DocFormat=rvddURL then
    StyleNo := rvActionInsertHyperlink1.GetHyperlinkStyleNo(
      Sender as TRichViewEdit);
end;
```

See also events:

- OnWriteHyperlink[418].

See also properties of RVRTFReadProperties[253]:

- BasePathLinks[459].

6.1.1.4.23 TRichView.OnReadMergeField

Occurs when reading content of merge fields from RTF file.

```
type

  TRVReadMergeFieldEvent =
    procedure (Sender: TCustomRichView;
      const FieldName, Extras: TRVUnicodeString[1099];
      DocFormat: TRVLoadFormat;
     var StyleNo: Integer; var ItemTag: TRVTag[1096];
     var ItemName: TRVUnicodeString[1099]) of object;

property OnReadMergeField: TRVReadMergeFieldEvent;
```

(introduced in version 18)

This event occurs when reading content of merge field from RTF. It allows modifying read content. This event is different from OnReadField[393], because:

- OnReadField occurs when reading field code and allows replacing field content, while OnReadMergeField occurs when reading field content and allows to modify it
- OnReadField occurs when reading DocX and RTF, OnReadMergeField occurs when reading RTF
- OnReadField occurs for any field unsupported by TRichView, OnReadMergeField occurs only for MERGEFIELD.

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[899])
- if the item is a text, it is an index of style[764] 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[96] of this item. You can store **FieldName** here. However, please note that targets of hyperlinks[97] 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[1099]; DocFormat: TRVLoadFormat[1081];
  var StyleNo: Integer; var ItemTag: TRVTag[1096];
  var ItemName: TRVUnicodeString[1099]);
begin
  if StyleNo >= 0 then
    ItemName := '{' + FieldName + '}';
end;
```

6.1.1.4.24 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[247])

```
type
  TRVDblClickEvent =
    procedure(Sender: TCustomRichView[216];
      ClickedWord: TRVUnicodeString[1099];
      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[90]). In this case **ClickedWord** contains a name of this item.

If **Style**>=0, then this is a text item[167], and **Style** is an index in the collection of text styles (Style[260].TextStyles[764]). 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[235] property.

This event is not generated when the user clicks not on an item (i.e. on a background)

See also events:

- OnRVRightClick[404].

See also properties:

- Delimiters[235].

6.1.1.4.25 TRichView.OnRVFControlNeeded

Occurs when reading inserted control [174] from RVF file or stream, and this control was saved without its "body".

```
type
  TRVFControlNeededEvent =
    procedure(Sender: TCustomRichView[216]; Name: TRVUnicodeString[1099];
      const Tag: TRVTag[1096]; 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 [254], none of control properties are saved. Only the item name (value returned by GetItemText [310]) and *tag* (value returned by GetItemTag [310]) 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 [310]; do not confuse with Name property of controls, see Item types [90]).

Tag – *tag* of the control's item (this is a value returned by GetItemTag [310]; do not confuse with Tag property of controls, see "Tags" [96]).

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 [129], contains example.

6.1.1.4.26 TRichView.OnRVFImageListNeeded

Occurs when reading *bullet* [176], *hotspot* [178] or list level with imagelist picture [863] from RVF or XML (by TRichViewXML).

```
type
  TRVFImageListNeededEvent =
    procedure(Sender: TCustomRichView[216];
      ImageListTag: Integer;
      var Ail: TCustomImageList) of object;

property OnRVFImageListNeeded: TRVFImageListNeededEvent;
```

If you do not process this event, TRichView is not able to read *bullets*[176], *hotspots*[178] and list level with imagelist from RVF file or stream, or XML files saved by TRichViewXML if its SaveImageLists property is equal to False.

When saving these items, TRichView saves Tag of the corresponding ImageList (do not confuse with RichView items' tags[96], this is TCustomImageList.Tag property). This value is passed as **ImageListTag** parameter of this event on reading.

Assign your ImageList to **Ail** 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/XML, so it is not recommended to perform time consuming calculations in this event.

See also:

- RVF overview[129], contains example.

6.1.1.4.27 TRichView.OnRVFPictureNeeded

Occurs when reading picture[169] or *hot-picture*[172] from RVF file or stream, and this picture was saved without its "body"

```
type
  TRVFPictureNeededEvent = procedure (Sender: TCustomRichView[216];
    const ItemName: TRVUnicodeString[1099]; Item: TRVNonTextItemInfo[899];
    Index1, Index2: Integer; var gr: TRVGraphic[1032]) 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[254], images are not saved. Only item name (value returned by GetItemText[310]) and *tag*[96] (value returned by GetItemTag[310]) 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[169] and hot-pictures[172];
- table[179] background images[904] (even if the saved table did not have a background image initially)
- table cell background images[953] (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[310]). 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[899].

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[129], contains an example.

6.1.1.4.28 TRichView.OnRVMouseDown

Occurs when the user presses a mouse button with the mouse pointer over the control.

```
type
  TRVMouseEvent =
    procedure(Sender: TCustomRichView[216]; Button: TMouseButton;
    Shift: TShiftState; ItemNo, X, Y: TRVCoord[1064]) 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[216];
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: TRVCoord[1064]);
var LRVData: TCustomRVFormattedData[1014];
    LItemNo, LOffs: Integer;
    pt: TRVCoordPoint[1064];
begin
  pt := MyRichView.ClientToDocument[286](Point(X,Y));
  if not MyRichView.GetItemAt[305](pt.X, pt.Y, LRVData, LItemNo, LOffs,
```

```
    True) then
    exit;
  // item is the LItemNo-th item in LRVData
  // example of using: LRVData.GetItemStyle (310) (LItemNo)
  ..
end;
```

Note: if you want to create code responding on clicking on item, it's recommended to use OnRVMouseUp (403) instead of OnRVMouseDown.

See also events:

- OnRVMouseUp (403).

6.1.1.4.29 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 (243), the next link has id=FirstJumpNo (243)+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 (483)) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly (488)=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 (390).

See also:

- RichView hypertext overview (97), contains example.

6.1.1.4.30 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 (216);
    Button: TMouseButton; Shift: TShiftState;
    ItemNo, X, Y: TRVCoord (1064)) 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[216];
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: TRVCoord[1064]);
var LRVData: TCustomRVFormattedData[1014];
    LItemNo, LOffs: Integer;
    pt: TRVCoordPoint[1064];
begin
  pt := MyRichView.ClientToDocument[286](Point(X,Y));
  if not MyRichView.GetItemAt[305](pt.X, pt.Y, LRVData, LItemNo, LOffs,
    True) then
    exit;
  // item is the LItemNo-th item in LRVData
  // example of using: LRVData.GetItemStyle[310] (LItemNo)
  ..
end;
```

See also events:

- OnRVMouseDown[402];
- OnRVRightClick[404].

6.1.1.4.31 TRichView.OnRVRightClick

Occurs when the user releases the right mouse button over the RichView component.

```
type
  TRVRightClickEvent = procedure(Sender: TCustomRichView[216];
    ClickedWord: TRVUnicodeString[1099]; Style, X, Y: TRVCoord[1064]) of object;

property OnRVRightClick:TRVRightClickEvent;
```

This event is almost obsolete. There are more convenient events:

- OnRVMouseDown[402],
- OnRVMouseUp[403].

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*, *rvsBreak*, *rvsHotspot* or *rvsBullet*[1127]). In this case **ClickedWord** contains name of this item (see Item types[90]).

If **Style**>=0, this is a text item, and **Style** is an index in the collection of text styles (Style[260].TextStyles[764]). 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[235] 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[402], OnRVMouseUp[403]):

- 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[399].

See also properties:

- Delimiters[235].

6.1.1.4.32 TRichView.OnSaveComponentToFile

Occurs when TRichView wants to save inserted control[174] in text, Markdown, RTF, HTML, or DocX file (or stream).

```
type
  TRVSaveComponentToFileEvent = procedure(
    Sender: TCustomRichView[216]; Path: TRVUnicodeString[1099];
    SaveMe: TPersistent; SaveFormat: TRVSaveFormat[1086];
    var OutStr:TRVUnicodeString[1099]) of object;

property OnSaveComponentToFile: TRVSaveComponentToFileEvent;
```

By default, inserted controls are not saved in HTML, RTF, DocX, Markdown, 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, rvsfMarkdown*.

Path – path where the file is saved. May be used, for example, for saving images (see SavePicture[349] method).

Additional input parameters

Sender. Style[260].RVData, **Sender.**Style[260].ItemNo identify the item to save. RVData – document containing item that is being saved; it can be **Sender**.RVData[254], table cell[951], 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[1099];
  SaveMe: TPersistent; SaveFormat: TRVSaveFormat;
  var OutStr: TRVUnicodeString[1099]);
var bmp: TBitmap;
  ImageFileName: TRVUnicodeString[1099];
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[411];
- Saving and loading[127];
- Export to HTML[132].

6.1.1.4.33 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[1099])
   of object;

property OnSaveDocXExtra: TRVSaveDocXExtraEvent;
```

(changed in version 18)

This event occurs when saving DocX (SaveDocX[341], SaveDocXToStream[342]).

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[764] collection, **Obj** is a text style object[821]	<w:rPr>	document files*
rv_docxs_ParaStyle	Occurs when saving a paragraph style. **Index1** is an index of the paragraph style in ParaStyles[757] collection, **Obj** is RVData[1014] containing this paragraph. **Index2** is an index of the first item of this paragraph.	<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[755].
	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[841]
		if **Index2** = -1:
		<w:abstractNum>;
		if **Index2** >= 0:
		<w:lvl>
			word/numbering.xml

Area	Meaning	Parent XML Element	XML file
rv_docxs_StyleTemplate	Occurs when saving a style template as a style sheet item. **Index1** is an index of the style template in StyleTemplates [762]. **Obj** is a style template [843].	<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 [762]. **Obj** is a style template [843].	<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 [762]. **Obj** is a style template [843].	<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 [901].	<w:tcPr>	document files*
rv_docxs_RowProps	Occurs when saving table row properties. **Index1** = row index, **Obj** is the table [901].	<w:trPr>	document files*
rv_docxs_TableProps	Occurs when saving table properties. **Obj** is the table [901].	<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 [237].

See also events:

- OnSaveRTFExtra [414].
- OnSaveHTMLExtra [409].

6.1.1.4.34 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[1099]) of object;

property OnSaveHTMLExtra: TRVSaveHTMLExtraEvent;
```

(changed in version 18)

This event occurs when saving HTML.

If HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstNormal*, **CSSVersion** parameter = True.

If HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstSimplified*, **CSSVersion** parameter = False.

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[1099]);
begin
  case Area of
  rv_thms_Head:
    HTMLCode := '<script></script>';
  rv_thms_BodyAttribute:
    HTMLCode := 'alink=#ff0000';
  rv_thms_Body:
    HTMLCode := '<P>This document is generated by '+
```

```
    '<A href="https://www.trichview.com">RichView</A></P>';
  end;
end;
```

Note: TRVHTMLSaveArea is defined in RVStyle.pas.

See also events:

- OnHTMLSaveImage [383];
- OnSaveImage2 [410];
- OnSaveParaToHTML [413];
- OnSaveRTFExtra [414];
- OnSaveDocXExtra [407].

See also:

- Export to HTML [132].

6.1.1.4.35 TRichView.OnSaveImage2

Occurs when TRichView saves image to HTML, Markdown, or XML (by TRichViewXML component) file or stream.

```
type
  TRVSaveImageEvent2 = procedure (Sender: TCustomRichView;
    Graphic: TRVGraphic[1032]; SaveFormat: TRVSaveFormat[1086];
    const Path, ImagePrefix: TRVUnicodeString[1099];
    var ImageSaveNo: Integer; var Location: TRVUnicodeString[1099];
    var DoDefault: Boolean) of object;

property OnSaveImage2: TRVSaveImageEvent2;
```

(introduced in version 1.8; "hidden" parameters are added in version 14; changed in version 18)

For HTML, this event is alternative to OnHTMLSaveImage [383]. 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.

For HTML, **OnSaveImage2** is not called if:

- *rvhtmlsioUseItemImageFileNames* is included in HTMLSaveProperties [244].ImageOptions [442] and value of *rvespImageFileName* item property [1072] is not empty;
- DoDefault parameter of OnHTMLSaveImage [383] was set to *False* for this image.

For Markdown, **OnSaveImage2** is not called if:

- *rvmdsoUseItemImageFileNames* is included in the MarkdownProperties [245].SaveOptions [456] and value of *rvespImageFileName* item property [1072] is not empty.

Input parameters:

Graphic – image to save.

SaveFormat is equal to *rvsfHTML, rvsfMarkdown, or rvsfXML*.

Path – destination directory of HTML or Markdown file (or **Path** parameter of SaveHTMLToStream [346]/SaveMarkdownToStream [349])

ImagePrefix – recommended beginning of image file name (HTMLSaveProperties[244].ImagesPrefix[443] or MarkdownProperties[245].ImagesPrefix[453])

ImageSaveNo – value that you can use to create unique file names (see GetNextFileName method in the description of OnHTMLSaveImage[383]).

"Hidden" input parameters:

Sender.Style.RVData (must be typecasted to TCustomRVData[1014]) – document containing the item that is being saved to an image; it can be **Sender**.RVData[254], table cell[951], 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*[176], *hotspots*[178], list markers[180] with images), TRichView creates a temporal TBitmap and passes it to this event.

See also events:

- OnHTMLSaveImage[383];
- OnSaveItemToFile[411];
- OnImportPicture[386].

See also:

- Export to HTML[132].

6.1.1.4.36 TRichView.OnSaveItemToFile

Allows you to change how document items are saved in text, RTF, DocX, HTML or Markdown file or stream.

```
type
  TRVSaveItemToFileEvent = procedure (Sender: TCustomRichView;
    const Path: TRVUnicodeString[1099];
    RVData: TCustomRVData[1014]; ItemNo: Integer;
    SaveFormat: TRVSaveFormat[1086]; var OutStr: TRVUnicodeString[1099];
    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[383],
- OnSaveImage2[410],
- OnSaveComponentToFile[405].

These events are easier to use than this event.

Input parameters:

Path – path to the output file;

RVData, **ItemNo** – item to save. **RVData** – document containing item that is being saved; it can be **Sender**.RVData [254], table cell [951], 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 text items	For non-text items
True	For text files: default saving, value of **OutStr** is ignored. For HTML, RTF, DocX, Markdown: 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 [183] only partially

Notes about Markdown:

Regardless of **DoDefault** value, space characters in **OutStr** can be encoded (changed to ' ' or a line break). To prevent it, use #01 instead of space characters (#01 will be changed to spaces before saving).

Example: saving pictures as '[PIC]'

```
procedure TMyForm.MyRichViewSaveItemToFile(
```

```
  Sender: TCustomRichView; const Path: TRVUnicodeString[1099];
  RVData: TCustomRVData[1014]; ItemNo: Integer;
  SaveFormat: TRVSaveFormat[1086];
  var OutStr: TRVUnicodeString[1099]; var DoDefault: Boolean);
begin
  if (RVData.GetItemStyle[310](ItemNo)=rvsPicture[1127]) then
  begin
    case SaveFormat of
      rvsfMarkdown:
        OutStr := '\[PIC\]';
      else
        OutStr := '[PIC]';
    end;
    DoDefault := False;
  end;
end;
```

6.1.1.4.37 TRichView.OnSaveParaToHTML

Allows adding your HTML code when saving paragraphs to HTML.

```
property OnSaveParaToHTML: TRVSaveParaToHTMLEvent

TRVSaveParaToHTMLEvent =
  procedure (Sender: TCustomRichView;
    RVData: TCustomRVData[1014]; ItemNo: Integer;
    ParaStart, CSSVersion: Boolean;
    var HTMLCode: TRVUnicodeString[1099]) 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[244] for the last paragraph).

RVData can be **Sender**.RVData[254], or table cell, or cell inplace editor's RVData[254].

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[1099]);
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[409];
- Export to HTML[132].

6.1.1.4.38 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[1099]) of object;

property OnSaveRTFExtra: TRVSaveRTFExtraEvent;
```

(changed in version 18)

This event occurs when saving RTF (SaveRTF[350], SaveRTFToStream[351]).

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[764] collection, **Obj** is a text style object[821] If **InStyleSheet**=*True*, this style is saved as a part of style sheet (when RTF is saved with *rvrtfSaveStyleSheet*[252] option)
rv_rtfs_ParaStyle	Occurs when saving paragraph style. **Index1** is an index of the paragraph style in ParaStyles[757] collection, **Obj** is a paragraph style object[828] If **InStyleSheet**=*True*, this style is saved as a part of style sheet (when RTF is saved with *rvrtfSaveStyleSheet*[252] option)

Area	Meaning
rv_rtfs_ListStyle	Occurs when saving list style and list levels. **Index1** is an index of the list style in ListStyles [755]. **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 [841]
rv_rtfs_StyleTemplate	Occurs when saving a style template as a style sheet item. **Index1** is an index of the style template in StyleTemplates [762]. **Index2** = -1 **Obj** is a style template [843].
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 [845] = *rvstkParaText*. **Index1** is an index of the style template in StyleTemplates [762]. **Index2** = -1 **Obj** is a style template [843].
rv_rtfs_CellProps	Occurs when saving table cell properties. **Index1** = row index, **Index2** =column index, **Obj** is the table [901].
rv_rtfs_RowProps	Occurs when saving table row properties. **Index1** = row index, **Obj** is the table [901].
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[1056]);
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 [237].

See also events:

- OnSaveDocXExtra [407];
- OnSaveHTMLExtra [409].

See also examples:

- *https://www.trichview.com/forums/viewtopic.php?t=64*.

6.1.1.4.39 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 [355];
- Deselect [294];
- SetSelectionBounds [372];
- (and sometimes SelectionExists [356]).

See also:

- Selection in RichView [112].

6.1.1.4.40 TRichView.OnSpellingCheck

Allows to mark misspelled words.

```
type
  TRVSpellingCheckEvent = procedure (Sender: TCustomRichView;
    const AWord: TRVUnicodeString[1099]; 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

StyleNo is a style of text item [167] containing this word (index in the collection **Sender**.Style [260] .TextStyles [764]).

Output parameter:

Misspelled, must be set to *True* if this word is misspelled.

Example (using Addict 3 spelling checker):

```
procedure TMyForm.MyRichViewEditSpellingCheck(Sender: TCustomRichView;
  const AWord: TRVUnicodeString[1099]; StyleNo: Integer;
  var Misspelled: Boolean);
```

```
begin
  Misspelled := not MyRVAddictSpell3.WordAcceptable(AWord);
end;
```

This event is called in thread context.

Using with TRVSpellChecker [728]

You have two options:

- register this TRichView component using RVSpelllChecker.RegisterEditor [736] method (and this event will be processed automatically), or
- process this event and use RVSpelllChecker.IsWordValid [735] method.

FireMonkey note:

If the current platform provides a spelling checking service, TRichViewEdit can use it (see CheckSpelling [480] property). In this case, checking does not require this event (however, if this event is assigned, it has a higher priority than a platform service).

See also:

- Live spelling check in TRichView [138].
- OnGetSpellingSuggestions [382] [FMX]

6.1.1.4.41 TRichView.OnStyleTemplatesChange

Occurs when the collection Style [260].StyleTemplates [762] was changed.

```
property OnStyleTemplatesChange: TNotifyEvent;
```

(introduced in version 14)

This event may be called only if UseStyleTemplates [263]=*True*.

It is called:

- when loading or inserting RVF, RTF, DocX files or streams (and StyleTemplateInsertMode [260] <>*rvstimIgnoreSourceStyleTemplates*);
- (in editor) by ChangeStyleTemplates [508] method;
- by RichViewActions that can modify this collection (TrvActionNew, TrvActionOpen, TrvActionInsertHyperlink, and actions calling ChangeStyleTemplates [508]: TrvActionStyleTemplates, TrvActionAddStyleTemplate).

6.1.1.4.42 TRichView.OnTextFound

Occurs when TRichView.SearchText[354] (or TRichViewEdit.SearchText[553]) finds a text.

```
type
  TRVTextFoundEvent = procedure(Sender: TCustomRichView[216];
    const AText: TRVUnicodeString[1099]; ARVData: TCustomRVData[1014];
    StartItemNo, StartItemOffs, EndItemNo, EndItemOffs: Integer;
    Invert: Boolean; var DoDefault: Boolean) of object;

property OnTextFound: TRVTextFoundEvent;
```

(introduced in version 19)

Parameters:

AText – text that was searched for.

ARVData – document containing this text; it can be **Sender**.RVData[254], a table cell[951], or RVData of an cell inplace-editor.

StartItemNo, StartItemOffs, EndItemNo, EndItemOffs define text position in **ARVData**. **StartItemNo** and **StartItemOffs** define the starting position (closer to the beginning of the document), **EndItemNo** and **EndItemOffs** define the ending position (closer to the end of the document). **StartItemNo** and **EndItemNo** are indexes of items of **ARVData**, **StartItemOffs** and **EndItemOffs** are positions in these items (see GetSelectionBounds[320] for additional information).

If **Invert** = *False*, the text should be selected from start to end. If **Invert** = *False*, the text should be selected from end to start (when searching backward)

If you leave **DoDefault** = *True*, the text will be selected. If you assign *False*, nothing will be done (but still SearchText will return *True*); in this case you can call StoreSearchResult[375] to store the found position (to continue search from it).

Possible applications for this event:

- Searching in unformatted documents. Sometimes you just need to know if a document contains the specified text. A selection requires a formatted document; so, if you forbid selection, you can search in unformatted text. This is especially useful for searching in multiple documents.
- Searching in TRVReportHelper[721].RichView[724]
- Implementing additional search conditions that you can verify in this event. For example, you can search only in text that has specific attributes.

6.1.1.4.43 TRichView.OnWriteHyperlink

Occurs when TRichView wants to save hypertext link to HTML, Markdown, RTF or DocX file

```
type
  TRVWriteHyperlink = procedure (Sender: TCustomRichView;
     id: Integer; RVData: TCustomRVData[1014]; ItemNo: Integer;
     SaveFormat: TRVSaveFormat[1086];
     var Target, Extras: TRVUnicodeString[1099]) of object;

property OnWriteHyperlink: TRVWriteHyperlink;
```

(introduced in v1.8; changed in version 18)

By default, items tags [96] are saved as targets of hypertext links in RTF, DocX, HTML, and Markdown.

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 [254], or table cell, or cell inplace editor's RVData [254])

ItemNo – index of the hyperlink-item inside **RVData**.

SaveFormat can be one of: *rvsfHTML, rvsfRTF, rvsfDocX, rvsfMarkdown.*

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 [96] as a target.

```
procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString[1099]);
begin
  Target := RVData.GetItemTag[310](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[1099]);
begin
  Target := RVData.GetItemTag[310](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[1072] automatically.

Example 3:

```
procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString[1099]);
begin
  case id-FirstJumpNo[243] 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[395].

See also methods:

- SaveHTML[342];
- SaveMarkdown[348];
- SaveRTF[350];
- SaveDocX[341];
- (and methods saving these formats to streams).

See also:

- Hypertext[97];
- Saving to HTML[132].

6.1.1.4.44 TRichView.OnWriteObjectProperties

Occurs when TRichView saves non-text objects to RTF, DocX, or HTML; allows saving additional item properties.

```
type // Defined in RVStyle unit
  TRVObjectExportProperties = record
    Id, Name: TRVUnicodeString;
  end;

type
  TRVWriteObjectPropertiesEvent = procedure(Sender: TCustomRichView[216];
    RVData: TCustomRVData[1014]; ItemNo: Integer;
```

```
    SaveFormat: TRVSaveFormat[1086];
    var Props: TRVObjectExportProperties) of object;

property OnWriteObjectProperties: TRVWriteObjectPropertiesEvent;
```

(introduced in version 19)

This event occurs when saving objects of the following types:

- pictures[169] and hot-pictures[172];
- *bullets*[176] and *hotspots*[178];
- list markers[180] containing images;
- equations[193];
- shapes (in Report Workshop).

All these items are exported as pictures.

Input parameters:

RVData – a document containing an item to save (it may be **Sender**.RVData[254], or table cell, or cell inplace editor's RVData[254])

ItemNo – an index of this item inside **RVData**.

SaveFormat can be *rvsfHTML* or *rvsfRTF* or *rvsfDocX*

Props contains initial values of properties.

Output parameters:

Props contains values of properties that will be saved in the file.

Properties for saving

Props.Id contains an object identifier. If specified, it must be unique within a document.

Format	Initial value of Id	Requirements for Id
rvsfRTF	'' (empty string)	If empty string, the identifier is not saved. Otherwise, it must be an integer value encoded as a string. We highly recommend using negative numbers for RTF image identifiers (zero or small positive values may result corrupted picture).
rvsfDocX	An integer number encoded as a string. This number is auto-generated during DocX saving, and may be changed on each saving.	The identifier must be unique integer value encoded as a string.

Format	Initial value of Id	Requirements for Id
rvsfHTML	Occurs when saving list style and list levels. **Index1** is an index of the list style in ListStyles [755]. **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 [841]	If empty string, the identifier is not saved. Otherwise, it must contains a string that begins with a letter (['A'..'Z','a'..'z']) and may be followed by any number of letters, digits (['0'..'9']), hyphens ('-'), underscores ('_'), colons (':'), and periods ('.').

Props.Name contains an object name.

Format	Initial value of Name	Requirements for Name
rvsfRTF	'' (empty string)	Name is not saved, ignored.
rvsfDocX	'Pic *N*', where N is the initial value of Prop.Id.	Any string without line breaks.
rvsfHTML	'' (empty string)	Name is not saved, ignored.

6.1.1.5 Classes of properties

- TRVDocParameters [422] is a type of TRichView [216].DocParameters [237] property.It defines document properties like page size, margins, etc.
- TRVDocumentProperty [429] is a type of TRichView [216].Document [238] property. It is used for LiveBinding (in Delphi XE2+).
- TRVMarkdownProperties [447] is a type of TRichView [216].MarkdownProperties [245]. It defines properties for reading and writing Markdown files and streams.
- TRVMarkdownDefaultItemProperties [445] is a type of TRichView [216].MarkdownProperties [245] .DefaultItemProperties [452]. It contains default properties for objects read from Markdown files and streams.
- TRVRTFReaderProperties [457] is a type of TRichView [216].RTFReadProperties [253] property. It defines properties for reading RTF and DocX files and streams.

6.1.1.5.1 TRVDocParameters

This is a type of TRichView [216].DocParameters [237]. It contains document's and page properties that can be written to or read from RVF [129] (RichView Format) and RTF (Rich Text Format). The components do not use these properties, but they can be assigned to TRVPrint [696] using the method AssignDocParameters [708].

Unit [VCL/FMX] RVDocParams / fmxRVDocParams.

Syntax

```
TRVDocParameters = class(TPersistent)
```

(introduced in version 10)

Properties

This class has the following properties:

- Units [428] – measurement units for margins, paper size, header and footer positions;
- PageWidth [426], PageHeight [426] – page size;
- Orientation [426] – page orientation;
- LeftMargin [425], RightMargin [426], TopMargin [427], BottomMargin [424] – margins;
- MirrorMargins [426] allow to exchange left and right margins on even pages;
- HeaderY [425], FooterY [425] – position of header and footer;
- TitlePage [427] shows/hides header and footer for the first page;
- FacingPages [424] shows/hides header and footer for even pages;
- ZoomMode [428] – zooming mode;
- ZoomPercent [428] – zooming percent.
- Author [424], Title [427], Comments [424] – text properties.

Methods

This class has the following methods:

- Reset [429] resets all properties to default values;
- ResetLayout [429] resets all sizes to default values measured in Units [428];
- UnitsPerInch [429] allows to convert between measurement units;
- ConvertToUnits [428] converts all sizes to the specified units.

6.1.1.5.1.1 Properties

In TRVDocParameters

- Author [424]
- BottomMargin [424]
- Comments [424]
- FacingPages [424]
- FooterY [425]
- HeaderY [425]
- LeftMargin [425]
- MirrorMargins [426]
- Orientation [426]
- PageHeight [426]
- PageWidth [426]
- Reset [429]
- ResetLayout [429]
- RightMargin [426]
- Title [427]
- TitlePage [427]
- TopMargin [427]
- Units [428]
- UnitsPerInch [429]

- ZoomMode[428]
- ZoomPercent[428]

Author of the document

```
property Author: TRVUnicodeString[1099];
```

(introduced in version 13; changed in version 18)

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd[555].

For RTF and DocX, and for HTML import, this property is used optionally (can be turned on/off using options). But for HTML saving, this property is always used.

Default value:

'' (empty string)

Bottom margin, measured in Units[428].

```
property BottomMargin: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.Margins[705].Bottom.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

Default value:

1.0

Comments to the document (document description)

```
property Comments: TRVUnicodeString[1099];
```

(introduced in version 13; changed in version 18)

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd[555].

For RTF and DocX, and for HTML import, this property is used optionally (can be turned on/off using options). But for HTML saving, this property is always used (to save HTML description).

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[129] (RichView Format), RTF (Rich Text Format) and DocX files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.FacingPages[703].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Headers and footers are assigned to TRichView using SetHeader[363] and SetFooter[362] methods.

Default value:

False

See also:

- TitlePage[427]

Distance from the bottom of page to the bottom of footer, measured in Units[428].

```
property FooterY: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.FooterY[704].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

Default value:

0.5

Distance from the top of page to the top of header, measured in Units[428].

```
property HeaderY: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.HeaderY[705].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

Default value:

0.5

Left margin, measured in Units[428].

```
property LeftMargin: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.Margins[705].Left.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

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[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.MirrorMargins[706].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

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[129] (RichView Format) and RTF (Rich Text Format) files.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Default value:

poPortrait

Page height, measured in Units[428].

```
property PageHeight: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

Default value:

11.692913386 (A4 height in inches)

Page width, measured in Units[428].

```
property PageWidth: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

Default value:

8.2677165354 (A4 width in inches)

Right margin, measured in Units[428].

```
property RightMargin: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.Margins[705].Right.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

Default value:

1.25

Title of the document

```
property Title: TRVUnicodeString[1099];
```

(introduced in version 13; change version 18)

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd[555].

For RTF and DocX, and for HTML import, this property is used optionally (can be turned on/off using options). But for HTML saving, this property is always used.

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[129] (RichView Format), RTF (Rich Text Format) and DocX files. It also affects saving headers and footers in HTML files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.TitlePage[706].

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

Headers and footers are assigned to TRichView using SetHeader[363] and SetFooter[362] methods.

Default value:

False

See also:

- FacingPages[424]

Top margin, measured in Units[428].

```
property TopMargin: TRVLength[1081];
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[708] assigns this property to TRVPrint.Margins[705].Top.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[555].

Default value:

1.0

Measurement units for all sizes in TRVDocParameters.

```
property Units: TRVUnits[1099];
```

Assignment to this property does not change other properties. Use ConvertToUnits[428] to convert all sizes to new units.

If **Units** = *rvuPixels*, one unit = 1/Style[260].UnitsPixelsPerInch[765] of an inch (or 1/96, if Style[260] = *nil*).

Use UnitsPerInch[429] to convert values between units.

In TRichViewEdit[469], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[555].

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[129] (RichView Format) and RTF (Rich Text Format) files.

Default value:

rvzmCustom

View scale, % (if ZoomMode[428]=*rvzmCustom*).

```
property ZoomPercent: Integer;
```

This property is not used by TRichView components, but it can be written and read from RVF[129] (RichView Format), DocX, and RTF (Rich Text Format) files.

Default value:

100

6.1.1.5.1.2 Methods

In TRVDocParameters

ConvertToUnits[428]
Reset[429]
ResetLayout[429]
UnitsPerInch[429]

Converts all sizes to **AUnits**.

```
procedure ConvertToUnits(AUnits: TRVUnits[428]);
```

This method converts all size properties from Units[428] to **AUnits**, then assigns Units[428] = **AUnits**.

The following properties are converted:

- PageWidth[426], PageHeight[426] – page size;
- LeftMargin[425], RightMargin[426], TopMargin[427], BottomMargin[424] – margins;

- HeaderY[425], FooterY[425] – position of header and footer.

Resets all properties to default values.

```
procedure Reset;
```

Resets all sizes to default values measured in Units[428].

```
procedure ResetLayout;
```

It makes sense to call this method on "File | New" command.

The following properties are reset:

- PageWidth[426], PageHeight[426] – page size;
- LeftMargin[425], RightMargin[426], TopMargin[427], BottomMargin[424] – margins;
- HeaderY[425], FooterY[425] – position of header and footer.

In addition, this method resets Orientation[426] to *poPortrait* and MirrorMargins[426] to *False*.

Returns the number of **Units** in one inch.

```
function UnitsPerInch(Units: TRVUnits[428]): TRVLength[1081];
```

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[428].

6.1.1.5.2 TRVDocumentProperty

This is a type of TRichView[216].Document[238]. This property is used for LiveBindings.

Unit [VCL/FMX] RichView / fmxRichView.

Syntax

```
TRVDocumentProperty = class (TPersistent)
```

(introduced in version 21)

You can use LiveBindings to link this property to a dataset field of TBlobField type.

This is an alternative way of using TRichView components to view and edit database fields, which works both in VCL and FireMonkey frameworks. Another way is using TDBRichView[604] and TDBRichViewEdit[616] components (available in VCL and LCL frameworks).

This property contains sub-properties controlling this linking.

Important note: in order to link **Document** to TBlobField, DBRV unit (or fmxDBRV unit for FireMonkey) must be included in the project (add it in "uses").

6.1.1.5.2.1 Properties

In TRVDocumentProperty

- AllowMarkdown[430]

- AutoDeleteUnusedStyles 430
- CodePage 431
- FieldFormat 431
- IgnoreEscape 432

Turn on/off Markdown loading.

```
property AllowMarkdown: Boolean;
```

(introduced in version 23)

This property controls loading from TBlobField linked to this TRichView using LiveBindings (Delphi XE2+)

If the field does not contain RVF, RTF, DocX, or HTML, field content will be loaded:

- as Markdown, if (**AllowMarkdown** = *True*) or (FieldFormat 431 = *rvdbMarkdown*)
- as a plain text, otherwise

Default value:

False

See also

- CodePage 431
- TDBRichView 604.AllowMarkdown 608
- TDBRichViewEdit 616.AllowMarkdown 621

If this property is set to *True*, the component deletes unused text 764, paragraph 757 and list 755 styles before loading.

```
property AutoDeleteUnusedStyles: Boolean;
```

This property controls loading from TBlobField linked to this TRichView using LiveBindings (Delphi XE2+)

If FieldFormat 431 = *rvdbRTF, rvdbDocX, rvdbHTML*, unused styles are auto-deleted even if **AutoDeleteUnusedStyles** = *False*.

It's highly recommended to assign *True* to this property. However, the following conditions are required to auto-delete styles correctly:

- this component must not share TRVStyle 739 component with other TRichView controls (others TRichView controls must not be linked to the same TRVStyle component via Style 260 property) and
- if the owner of this property is TRichViewEdit 469, and documents are created in this editor and saved as RVF 113, the collections of text, paragraph and list styles must be saved together with documents (see RVFOptions 254, *rvfoSaveTextStyles* and *rvfoSaveParaStyles* must be included for **AutoDeleteUnusedStyles** = *True*).

Default value:

False

See also

- TDBRichView[604].AutoDeleteUnusedStyles[608]
- TDBRichViewEdit[616].AutoDeleteUnusedStyles[621]

Encoding for saving and loading plain text and Markdown.

```
property CodePage: TRVCodePage[1062];
```

(introduced in version 23)

This property controls loading and saving from/to TBlobField linked to/from this TRichViewEdit[469] using LiveBindings (Delphi XE2+).

The default value of this property is *RV_CP_DEFAULT*[1118]. See the topic about this constant for details.

This property is not used when working with WideMemo fields (for these fields, text and Markdown are always saved/loaded as Unicode (UTF-16))

Default value:

RV_CP_DEFAULT[1118]

See also

- TDBRichView[604].CodePage[609]
- TDBRichViewEdit[616].CodePage[622]
- TRichViewEdit[469].OnSaveCustomFormat[595]

Format for saving data in the database field (linked using LiveBindings)

```
property FieldFormat: TRVDBFieldFormat[1064];
```

This property controls loading from TBlobField linked to this TRichViewEdit[469] using LiveBindings (Delphi XE2+). This property is used only in TRichViewEdit (because TRichView does not save data to a database field).

This property allows changing the field saving format to RVF (RichView Format)[129], RTF, DocX, HTML, Markdown or plain text.

Markdown and a plain text are saved using the specified CodePage[431] (unless saving to WideMemo fields; in these fields, text and Markdown are always saved in UTF-16 encoding).

Note: It is not guaranteed that document saved in RTF, DocX, HTML and text will be the same after reloading (some item types, text and paragraph attributes cannot be saved in RTF, DocX, HTML or plain text). To store all attributes of text and objects, use RVF format.

You can save data in your own format using OnSaveCustomFormat[595] event.

Default value:

rvdbRVF

See also

- TDBRichViewEdit[616].FieldFormat[623]
- TRichViewEdit[469].OnSaveCustomFormat[595]

Disallows processing of Escape key

```
property IgnoreEscape: Boolean;
```

This property controls loading document from TBlobField linked to this TRichViewEdit[469] using LiveBindings (Delphi XE2+). This property is used only in TRichViewEdit (because TRichView does not save data to a database field).

By default, Escape cancels editing (the linked data are reset to the last saved version).

If you set this property to *True*, Escape will do nothing.

Default value:

False

See also:

- TDBRichViewEdit[616].IgnoreEscape[624]

6.1.1.5.3 TRVHTMLReaderProperties

This is a type of TRichView[216].HTMLReadProperties[243]. It contains properties for HTML import.

Do not create objects of this class yourself, use this property instead.

Unit [VCL/FMX] RVHtmlReadProps / fmxRVHtmlReadProps.

Syntax

```
TRVHTMLReaderProperties = class (TPersistent)
```

Properties

Allowing/disallowing loading some content:

- ReadBackground[434] – reading background and margins
- ReadDocParameters[435] – reading title, author, and description
- SkipHiddenText[435] – reading hidden content
- IDAsCheckpoints[434] – reading IDs (as checkpoints[92])

Options:

- EnforceListLevels[433] – choosing between keeping a list level / a list type
- FontSizePrecision[434] – font size precision
- XHTMLCheck[436] – turns XHTML stricter syntax checking on/off.

6.1.1.5.3.1 Properties

In TRVHtmlReaderProperties

- BasePathLinks[433]
- EnforceListLevels[433]
- FontSizePrecision[434]
- IDAsCheckpoints[434]
- ReadBackground[434]
- ReadDocParameters[435]

- SkipHiddenText[435]
- XHTMLCheck[436]

Specifies whether to add base path (path to HTML file) to hyperlink targets.

```
property BasePathLinks: Boolean;
```

(introduced in version 21)

Set to *False* if you do not want to add the document path to relative hyperlinks on HTML loading.

For example, when loading "https://www.trichview.com/index.html" containing link to "download.html":

- if this property is *False*, TCustomRichView.OnReadHyperlink[395] event occurs with Target='download.html' (or, if the event is not assigned, this value is stored in them item tag[96]);
- if this property is *True*, this event occurs with Target='https://www.trichview.com/download.html' (or, if the event is not assigned, this value is stored in them item tag).

Default value:

True

See also:

- TRichView.RTFReadProperties[253].BasePathLinks[459]
- *rvmdloBasePathLinks* option in TRichView.MarkdownProperties[245].LoadOptions[453]

This property specifies whether the components should choose to keep a list level or a list type.

```
property EnforceListLevels: Boolean;
```

(introduced in version 21)

In HTML model, all lists have only one level, and lists can be nested.

In TRichView (like in Microsoft Word), lists may have multiple levels but cannot be nested.

As a result, the following list cannot be loaded as it is (because it has different type of markers at the second level):

```
• itemA
  o itemAA
• itemA
   1. itemBA
```

If *rvmdloEnforceListLevels* is included, this list is loaded as:

```
• itemA (level 1)
  o itemAA (level 2)
• itemB (level 1)
  o itemBA (level 2)
```

If *rvmdloEnforceListLevels* is excluded, this list is loaded like the following list:

```
• itemA (level 1)
  o itemAA (level 2)
• itemB (level 1)
   1. itemBA (level 3)
```

Default value:

True

See also:

- *rvmdloEnforceListLevels* option in TRichView.MarkdownProperties [245].LoadOptions [453]

Defines the precision of font size calculation for text loaded from HTML

```
type // defined in RVStyle.pas
  TRVFontSizePrecision = (
    rvfpPoints,
    rvfpHalfPoints,
    rvfpMax
  );
property FontSizePrecision: TRVFontSizePrecision;
```

(introduced in version 21)

Value	Meaning
rvfpPoints	Precision: 1 point (font sizes are loaded as integer values)
rvfpHalfPoints	Precision: 1/2 of a point (font sizes like 10.5 points are possible)
rvfpMax	VCL and LCL: the same as *rvfpHalfPoints* FireMonkey: maximum prevision, font sizes may be loaded as fractional values.

Default value:

rvfpHalfPoints

Specifies whether "id" attributes of HTML tags are loaded as checkpoints [92].

```
property BasePathLinks: Boolean;
```

(introduced in version 21)

In HTML documents, link destination anchors may be created using:

- <a> tag with "name" attribute
- any tag with "id" attribute

If **IDAsCheckpoints** = *True:* the both options are loaded as checkpoints.

If **IDAsCheckpoints** = *False:* only <a> tags are loaded as checkpoint.

Default value:

True

Allows reading document background and margins from HTML.

```
property ReadBackground: Boolean;
```

(introduced in version 21)

Background and margins can be read only when loading HTML; when inserting or pasting HTML, they are always ignored.

The result is assigned to the following properties of TRichView[216] control:
- BackgroundPicture[229]
- BackgroundProperties[229]
- LeftMargin[245], TopMargin[262], RightMargin[251], BottomMargin[231]

Default value:

True

Allows reading title, author, and description from HTML.

```
property ReadDocParameters: Boolean;
```

(introduced in version 21)

These properties can be read only when loading HTML; when inserting or pasting HTML, they are always ignored.

The result is assigned to the following properties of TRichView[216].DocParameters[237] control:
- Title[427] (from <title>)
- Comments[424] (from <meta name="description">'s content)
- Author[424] (from <meta name="author">'s content)

Default value:

False

This property controls how hidden text is loaded from HTML

```
property SkipHiddenText: Boolean;
```

(introduced in version 21)

If *True*, hidden content is not added to TRichView.

If *False*, hidden content[106] is added.

Default value:

True

See also:
- TRVRTFReaderProperties[457].SkipHiddenText[466]

Specifies when XHTML syntax checking is performed.

```
type // defined in RVStyle.pas
  TRVXHTMLCheck = (
    rvxhtmlcYes,
    rvxhtmlcNo
    rvxhtmlcAuto);

property XHTMLCheck: TRVXHTMLCheck;
```

(introduced in version 22)

XHTML has stricter syntax comparing to HTML. TRichView can perform more strict checking of XHTML, stoppling on errors such as unclosed tags.

This property specifies when such checking is done.

Value	Meaning
rvxhtmlcYes	Always
rvxhtmlcNo	Never
rvxhtmlcAuto	Only for files with "XML" and "XHTML" extensions.

Default value:

rvxhtmlcAuto

6.1.1.5.4 TRVHTMLSaveProperties

This is a type of TRichView[216].HTMLSaveProperties[244]. It contains properties for HTML export.

Do not create objects of this class yourself, use this property instead.

Unit [VCL/FMX] RVHtmlSaveProps / fmxRVHtmlSaveProps.

Syntax

```
TRVHTMLSaveProperties = class (TPersistent)
```

Properties

By default, TRichView saves HTML with formatting represented by CSS (Cascading Style Sheets). You can switch to simplified HTML (using formatting tags instead of CSS) using HTMLSavingType[441].
CSS can be customized using CSSOptions[438].
By default, CSS styles are saved as a style sheet in HTML <head>, but they can be saved inline (CSSSavingType[438]) or externally (ExternalCSSFileName[440]).
Additional CSS styles can be provided in ExtraStyles[440] property.

TRichView can write HTML or XHTML, depending on HTMLVersion[441]. Encoding is specified in Encoding[439].

Images can be saved in external files or directly in HTML, customized by ImageOptions[442], ImagesPrefix[442].
Checkpoints are saved as anchors, customized by UseCheckpointNames[445], CheckpointsPrefix[437].
List markers are saved as HTML lists or as text/images, depending on ListMarkerSavingType[444].

Header and footer can be saved in HTML, if specified in SaveHeaderAndFooter[445].

Note: the default value of ImageOptions[442] depends on the control type:
- for TRichView[216], TRichViewEdit[469], TDBRichView[604]: []
- for TDBRichViewEdit[616]: [*rvhtmlsioInlineImages*]

6.1.1.5.4.1 Properties

In TRVHTMLSaveProperties

- CheckpointsPrefix[437]
- CSSOptions[438]
- CSSSavingType[438]
- Default0Style[439]
- Encoding[439]
- ExternalCSSFileName[440]
- ExtraStyles[440]
- HTMLSavingType[441]
- HTMLVersion[441]
- ImageOptions[442]
- ImagesPrefix[443]
- ListMarkerSavingType[444]
- SaveHeaderAndFooter[445]
- UseCheckpointNames[445]

Specifies prefix for names of anchors in saved HTML.

```
property CheckpointsPrefix: TRVUnicodeString[1099];
```

(introduced in version 21)

This property is used to generate names of anchors in HTML.

It is used only if UseCheckpointNames[445] = *False*.

Default value:

'cp'

Options for saving CSS in HTML.

```
type // Defined in RVStyle.pas
  TRVHTMLCSSSaveOption = (
    rvcsssoForceNonTextCSS,
    rvcsssoNoDefCSSStyle
  );
  TRVHTMLCSSSaveOptions = set of TRVHTMLCSSSaveOption;
```

```
property CSSOptions: TRVHTMLCSSSaveOptions;
```

(introduced in version 21)

Option	Meaning
rvcsssoForceNonTextCSS	If set, CSS is used for saving some non-text items (such as images and tables), even if HTMLSavingType [441] = *rvhtmlstSimplified*.
rvcsssoNoDefCSSStyle	By default, TRichView assigns attributes of TextStyles[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[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>.

Default value:

[]

Specifies how CSS styles are saved in HTML.

```
type // defined in RVStyle.pas
  TRVCSSSavingType = (
    rvcssstStyleSheet,
    rvcssstInline
  );
```

```
property CSSSavingType: TRVCSSSavingType;
```

(introduced in version 21)

This property is used only if HTML is saved using CSS, i.e. HTMLSavingType [441] = *rvhtmlstNormal*.

Value	Meaning
rvcssstStyleSheet	Styles are saved as a style sheet (and referred in "class" attributes of tags). Recommended. By default, styles are saved in <style> tag. But if ExternalCSSFileName[440] is defined, it is used instead.
rvcssstInline	Styles are saved inline (in "style" attributes of tags). This option generates highly overbloated HTML. Not recommended. This option may be useful when generating HTML that will be used as a part of larger HTML file.

Default value:

rvcssstStyleSheet

Allows using default browser font.

```
property Default0Style: Boolean;
```

(introduced in version 21)

If *True*, no formatting is saved for text of the 0-th style (TextStyles[764][0] of the linked TRVStyle[739] component), so browsers will use a default font for it.

This option affects appearance of all styles, because only attributes different from attributes of the 0th style are saved.

Default value:

False

Encoding for saving HTML files.

```
property Encoding: TRVHTMLEncoding[1078];
```

(introduced in version 21)

Note: for WideMemo fields, TDBRichViewEdit[616] component stores HTML in UTF-16 encoding, ignoring **Encoding** and EncodingStr[439].

Default value:

rvhtmleUTF8

See also:

- EncodingStr[439]

Encoding string for writing in HTML files.

```
property EncodingStr: TRVUnicodeString[1099];
```

(introduced in version 21)

If **EncodingStr** is not empty, its value is written in saved HTML files to define encoding.

If **EncodingStr** empty, encoding value for writing in HTML is calculated automatically, basing on Encoding [439] property.

Unlike Encoding [439], **EncodingStr** does not affect how HTML is saved, it only modifies HTML header.

This property may be useful if you want to convert HTML to another encoding after saving.

Note: for WideMemo fields, TDBRichViewEdit [616] component stores HTML in UTF-16 encoding, ignoring **EncodingStr** and Encoding [439].

Example: how to save HTML in UTF-16 encoding

1. Assign Encoding [439] = *rvhtmleUTF8*, **EncodingStr** = 'utf-16'.

2. Save HTML.

3. Convert the result from UTF-8 to UTF-16.

Now you have HTML in UTF-16 encoding, with 'utf-16' written in its header.

Default value:
'' (empty string)

Specifies a location of CSS styles for writing to HTML.

```
property ExternalCSSFileName: TRVUnicodeString[1099];
```

(introduced in version 21)

This property is used only if HTML is saved using CSS, i.e. HTMLSavingType [441] = *rvhtmlstNormal*.

It allows saving CSS styles externally, instead of inserting them directly in HTML file.

This string is inserted inside <link rel="stylesheet"> tag of saved HTML files.

If this property is defined (not empty), TRichView does not generate style sheet itself (even if CSSSavingType [438] = *rvcssstStyleSheet*). In this case, it is assumed that this file is created using TRVStyle.SaveCSS [772] method.

Specifies additional CSS styles for writing to HTML.

```
property ExtraStyles: TRVUnicodeString[1099];
```

(introduced in version 21)

This string is inserted inside <style> tag of saved HTML files.

If TRichView saves CSS style sheet itself (it happens if HTMLSavingType [441] = *rvhtmlstNormal*, and CSSSavingType [438] = *rvcssstStyleSheet*, and ExternalCSSFileName [440] = ''), **ExtraStyles** are written after TRichView style sheet (so **ExtraStyles** have higher priority).

ExtraStyles are always saved, even if HTMLSavingType [441] = *rvhtmlstSimplified*.

Default value:

'' (empty string)

Specifies how text formatting and object properties are represented in saved HTML.

```
type // Defined in RVStyle.pas
  TRVHTMLSavingType =
    (rvhtmlstNormal, rvhtmlstSimplified);

property HTMLSavingType: TRVHTMLSavingType;
```

(introduced in version 21)

Value	Meaning
rvhtmlstNormal	Normal HTML: formatting is represented by CSS (Cascading Style Sheets)
rvhtmlstSimplified	Simplified HTML: formatting is represented by deprecated HTML tags (like <font> or <b>). CSS is not used. For non-text items (like images or tables) CSS still can be used, if *rvcsssoForceNonTextCSS* is included in CSSOptions [438].

Default value:

rvhtmlstNormal

Specifies the HTML specification to use for saved documents.

```
type // Defined in RVStyle.pas
  TRVHTMLVersion = (rvhtmlvHTML, rvhtmlvXHTML)

property HTMLVersion: TRVHTMLVersion;
```

(introduced in version 21)

Value	Meaning
rvhtmlvHTML	HTML 4
rvhtmlvXHTML	XHTML 1

Default value:

rvhtmlvHTML

Options for saving images in HTML.

```
type // Defined in RVStyle.pas
  TRVHTMLSaveImageOption = (
    rvhtmlsioInlineImages,
    rvhtmlsioUseItemImageFileNames,
    rvhtmlsioImageSizes,
    rvhtmlsioImageSizeAttributes,
    rvhtmlsioOverrideImages,
    rvhtmlsioSuppressLinkUnderline
    );
  TRVHTMLSaveImageOptions =
    set of TRVHTMLSaveImageOption;
```

```
property ImageOptions: TRVHTMLSaveImageOptions;
```

(introduced in version 21, changed in v22)

Option	Meaning
rvhtmlsioInlineImages	If set, images are stored directly in HTML file (in base64 encoding). If not set, images are stored in external files.
rvhtmlsioUseItemImageFileNames	If included, images with defined (non-empty) file names will not be saved, but their file names will be written in HTML. File names for items are defined in *rvespImageFileName* string property[1072]. This option has higher priority than *rvhtmlsioInlineImages*. See the explanation below this table.
rvhtmlsioImageSizes	If set, TRichView saves widths and heights of images in HTML explicitly, even for non-resized images. This option does not affect pictures having defined *rvepImageWidth* and/or *rvepImageHeight* item extra integer properties[1067]. Sizes are always saved for such images. You can override this setting for the specific item using *rveoNoHTMLImagesSize* item extra integer properties[1067].
*rvhtmlsioImageSizeAttributes**	If set, widths and height of images are always saved as attributes of <img>, not as CSS properties.

Option	Meaning
rvhtmlsioOverrideImages	This option is used if images are stored in external files (i.e., *rvhtmlsioInlineImages* is not included). If set, TRichView can override existing images. If not set, TRichView generates unique names for images.
*rvhtmlsioSuppressLinkUnderline**	If set, underlines for hyperlinks around images (and other non-text object) will be explicitly suppressed. This option works only if HTMLSavingType 441 = *rvhtmlstNormal*.

* These options are useful when saving HTML which will be used to create an email that can be opened in Microsoft Outlook. They are not needed if HTML will be saved for viewing in modern browsers.

When TRichView saves an image in HTML, the following steps are performed:

1. OnHTMLSaveImage 383 event occurs. If *rvhtmlsioUseItemImageFileNames* is included, *rvespImageFileName* string property 1072 is used as the initial value of image path.
   If the image is not saved in this event, then
2. If *rvhtmlsioUseItemImageFileNames* is included, and value of *rvespImageFileName* string property 1072 is not empty, it is written to HTML; the image file is not saved.
   Otherwise
3. OnSaveImage2 410 occurs.
   If the image is not saved in this event, then
4. If *rvhtmlsioInlineImages* is included, the image is saved directly in HTML (base64-encoded).
   Otherwise
5. The image is stored in an external file. A name of this file is auto-generated using *rvhtmlsioOverrideImages* option and ImagesPrefix 443 property.

On the two last steps, non-web images (such as icons or metafiles) are saved as Jpeg images.

Default value:

- for TRichView 216, TRichViewEdit 469, TDBRichView 604: []
- for TDBRichViewEdit 616, TcxTRichViewEdit.Properties: [*rvhtmlsioInlineImages*]

See also

- TRVMarkdownProperties 447.SaveOptions 456 (has similar options for saving images in Markdown)

Specifies prefix for names of image files.

```
property ImagesPrefix: TRVUnicodeString 1099;
```

(introduced in version 21)

This property is used to generate names of files for saving images included in saved HTML.

It is used when images are stored in external files (*rvhtmlsioInlineImages* is not included ImageOptions)[442].

Default value:

'img'

Defines how paragraph lists[180] are saved in HTML.

```
type // Defined in RVStyle.pas
  TRVHTMLListMarkerSavingType = (
    rvhtmlmstNormal,
    rvhtmlmstAsText
  );
```

```
property ListMarkerSavingType: TRVHTMLListMarkerSavingType;
```

(introduced in version 21)

Value	Meaning
rvhtmlmstNormal	Normal saving, using , , tags
rvhtmlmstAsText	Saving list markers as text or images (depending on the marker type).

rvhtmlmstNormal mode

When saving HTML using CSS (HTMLSavingType[441] = *rvhtmlstNormal)*, CSS of lists are inserted directly in HTML (in "style" attribute):

- *rvlstBullet, rvlstUnicodeBullet*[864] – if possible, exported as bullets with default markers (disc, circle or square); otherwise, as FormatString[861];
- *rvlstDecimal, rvlstLowerAlpha*[864] and other numbering – exported as lists with the proper numbering. FormatString[861] and Font[861] are ignored;
- *rvlstPicture, rvlstImageList, rvlstImageListCounter*[864] – exported as a bullet with picture. It's possible to change default image saving by OnHTMLSaveImage[383];
- indents are saved

When saving without CSS (HTMLSavingType[441] = *rvhtmlstSimplified)*:

- *rvlstBullet, rvlstUnicodeBullet, rvlstPicture, rvlstImageList*[864] – exported as bullets with default markers (disc, circle or square). FormatString[861], Picture[868], ImageList[863] and Font[861] are ignored;
- *rvlstDecimal, rvlstLowerAlpha*[864] and other numbering – exported as lists with the proper numbering (if this numbering type cannot be represented in HTML, as a decimal numbering). FormatString[861] and Font[861] are ignored;
- *rvlstImageListCounter*[864] – saved as a decimal numbering;
- indents are not saved.

rvhtmlmstAsText mode

Markers are saved without special HTML keywords for lists (like <ol>, <ul>, <li>).

Font[861] and format strings[861] settings are respected.

When saving HTML using CSS (HTMLSavingType[441] = *rvhtmlstNormal):*

LeftIndent[864] and MarkerIndent[866] are retained. FirstIndent[860] are retained for non-hanging markers only (MarkerIndent>LeftIndent+FirstIndent)..

When saving without CSS (HTMLSavingType[441] = *rvhtmlstSimplified)*: all indents are ignored.

Generally, HTML files saved with this option look closer to the original.

Default value:

rvhtmlmstNormal

Allows saving header and footer to HTML

```
property SaveHeaderAndFooter: Boolean;
```

(introduced in version 21)

If *True*, 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[363]/SetFooter[362] methods.

If DocParameters[237].TitlePage[427] = *True*, the component uses a header and a footer for the first page. Otherwise, a normal header and footer are used.

If CSS is used (HTMLSavingType[441] = *rvhtmlstNormal)*, headers and footers always use inline CSS (regardless of value of CSSSavingType[438]).

Default value:

False

Specifies how checkpoints are saved to HTML.

```
property UseCheckpointNames: Boolean;
```

(introduced in version 21)

If *True*, names of checkpoints[92] are used as HTML anchor names. They should be unique.

If *False*, anchors names are generated using CheckpointsPrefix[437] property.

Default value:

True

6.1.1.5.5 TRVMarkdownDefaultItemProperties

This is a type of TRichView[216].MarkdownProperties[245].DefaultItemProperties[452].

It contains values of properties assigned to items loaded from ***Markdown***.

Do not create objects of this class yourself, use this property instead.

Unit [VCL/FMX] RVMarkdown / fmxRVMarkdown.

Syntax

```
TRVMarkdownDefaultItemProperties = class (TPersistent)
```

Properties

Colors:

- BlockQuoteBackColor [446] – color of block quotes (if StyleTemplates are not used)
- BreakColor [446] – color of *breaks* [173]
- CodeBlockBackColor [446] – color of code blocks (if StyleTemplates are not used)

Sizes:

- ListLeftIndentPix, ListMarkerIndentPix [447] – indents of list items and block quotes
- TableCellWidthPix [447] – cell widths

Other properties:

- ImageVAlign [447] – alignment of images

6.1.1.5.5.1 Properties

In TRVMarkdownDefaultItemProperties

- BlockQuoteBackColor [446]
- BreakColor [446]
- CodeBlockBackColor [446]
- ImageVAlign [447]
- ListLeftIndentPix [447]
- ListMarkerIndentPix [447]
- TableCellWidthPix [447]

Background color of block quote paragraphs loaded from Markdown, if StyleTemplates are not used.

```
property BlockQuoteBackColor: TRVColor [1063];
```

This property is used only if StyleTemplates are not used [263], or 'Quote' and 'Intense Quote' StyleTemplates do not exist. Otherwise, block quote paragraphs are formatted according to the StyleTemplate.

Default value:
rvclNone [1105]

Color of thematic breaks [173] loaded from Markdown.

```
property BreakColor: TRVColor [1063];
```

If **BreakColor** = *clNone*, default color is used.

Default value:
rvclNone [1105]

Background color of code block paragraphs loaded from Markdown, if StyleTemplates are not used.

```
property CodeBlockBackColor: TRVColor [1063];
```

This property is used only if StyleTemplates are not used [263], or 'HTML Preformatted' StyleTemplate does not exist. Otherwise, code block paragraphs are formatted according to the StyleTemplate.

Default value:

rvclNone [1105]

Alignment of images loaded from Markdown.

```
property ImageVAlign: TRVVAlign[1100];
```

Default value:

rvvaBaseline

Indent of the list marker and list text (relative to the left indent of the previous list level)

```
property ListMarkerIndentPix: TRVPixel96Length[1085];
property ListLeftIndentPix: TRVPixel96Length[1085];
```

These properties are measured in pixels at 96 DPI. They are used to calculate values of MarkerIndent [866] and LeftIndent [864] of list levels, as well as left indents of block quotes (if StyleTemplates are not used [263], or 'Quote' and 'Intense Quote' StyleTemplates do not exist; otherwise, indents specified in the StyleTemplate is used).

MarkerIndent [866] is increased by **ListMarkerIndentPix** (converted to Units [764]).

LeftIndent [864] is increased by **ListMarkerIndentPix** + **ListLeftIndentPix** (converted to Units [764]).

Default value:

24

Widths of cells of tables loaded from Markdown.

```
property TableCellWidthPix: TRVPixel96Length[1085];
```

If this property has a positive value, it is measured in pixels at 96 DPI.

If this property has a negative value, it is measured in % of document width.

If this property is zero, cells have default widths.

Default value:

150

6.1.1.5.6 TRVMarkdownProperties

This is a type of TRichView [216].MarkdownProperties [245]. It contains properties for controlling ***Markdown*** export and import.

Do not create objects of this class yourself, use this property instead.

Unit [VCL/FMX] RVMarkdown / fmxRVMarkdown.

Syntax

```
TRVMarkdownProperties = class (TPersistent)
```

Properties

For exporting [448]**:**

- ImagesPrefix [453];
- LineWidth [453];
- NotesSavedAsFootnotes [455];

- SaveHiddenText [456];
- SaveOptions [456].

For importing [449]:

- DefaultItemProperties [452];
- LoadOptions [453].

Common:

- Extensions [452].

About Markdown

Markdown is a lightweight markup language with plain-text-formatting syntax.

TRichView implementation follows ***CommonMark specification***.

TRichView implementation of table extension follows ***GitHub Flavored Markdown specification***.

About Markdown export

TRichView can export to Markdown:

- text [167] (with italic and bold attributes)
- headings
- bullets and numbering [180]
- horizontal lines [173]
- tables [179] (optionally)
- footnotes [185], endnotes [184], sidenotes [186], text boxes [188] (optionally, as footnotes)
- pictures
- hyperlinks

Headings

Headings are determined by OutlineLevel [835] properties of paragraphs.

Markdown format supports two kinds of headings:

- Setext headings (paragraphs underlined with '=' (for heading 1) or '-' (for heading 2) characters)
- ATX headings (paragraphs started from '#' characters: '# ' for heading 1, '## ' for heading 2, ..., up to level 6).

When a Markdown file is read, only a heading level matters, Setext and ATX headings are loaded identically.

Setext headings can be multiline and may include hard line breaks but they can represent only headings of level 1 and 2. Note: some Markdown readers do not support multiline Setext headings.

See also *rvmdsoUseSetextHeadings* option in SaveOptions [456].

Text attributes

If StyleTemplates are not used [263], the following rules are applied when exporting text:

- for headings, bold and italic attributes are not saved
- for normal text, bold and italic attributes are taken from text formatting (Style [817] property)

If StyleTemplates are used [263], the following rules are applied when exporting text:

- if text is formatted using a style template named 'Emphasis', 'Subtle Emphasis', 'Intense Emphasis', it is saved as an emphasis

- if text is formatted using a style template named 'Strong', it is saved as a strong emphasis
- otherwise, formatting that is applied on top of paragraph style template is used (Style [817] property). For headings, a difference is calculated from 'heading #' style template; for normal text, a difference is calculated from 'Normal' style; for block quotes, a difference is calculated from the style template that is actually used for the text paragraph. For hyperlinks, 'Hyperlink' style template is taken into account as well.

Block quotes

If StyleTemplates are not used [263], block quotes are not saved.

If StyleTemplates are used [263], paragraphs formatted using a style template named 'Quote' or 'Intense Quote' are saved as block quotes.

TRichView supports exporting only a single level of block quotes.

See also *rvmdsoListInBlockQuote* option in SaveOptions [456].

Tables

The original Markdown format cannot contain tables. However, there is a Markdown extension that supports very simple tables (single-line text content, no cell merging, no nested tables, no additional table properties; content of columns can be left- or right- aligned or centered).

To enable saving tables as tables, include *rvmdeTables* in Extensions [452]. Otherwise, contents of cells are saved as normal paragraphs.

Notes

The original Markdown format cannot contain notes. However, there is a Markdown extension that supports footnotes.

To enable saving selected notes as footnotes, include *rvmdeFootnotes* in Extensions [452], and choose note types in NotesSavedAsFootnotes [455]. Otherwise, contents of notes are saved in place where they are inserted, as text inside [ ].

Encoding

TRichView exports Markdown in UTF-8 encoding.

It you want to convert output to some ANSI encoding, I suggest to change the value of the global typed constant RVMarkdownSmallSpace (defined in RVMarkdownSave unit) to space character ('#32;') or another space character that exists in the target encoding.

```
  RVMarkdownSmallSpace: TRVUnicodeString = '#8202;'; // UNI_HAIR_SPACE
```

TRichView may insert this code between a punctuation and a letter character to allow making the punctuation bold or italic.

About Markdown import

TRichView can import from Markdown:

- text [167] (with emphasis and strong emphasis, code spans)
- headings
- multilevel block quotes
- code blocks
- bullets and numbering [180]
- horizontal lines [173]

- tables [179] (optionally)
- pictures
- hyperlinks

StyleTemplates

If StyleTemplates are used [263], imported text and paragraph attributes depend on them.

StyleTemplates of the following names are used, if exist:

- 'Normal' (normal paragraphs)
- 'heading 1' ... 'heading 2' (heading paragraphs)
- 'Quote' or 'Intense Quote' (block quote paragraphs)
- 'HTML Preformatted' (code block paragraphs)
- 'HTML Code' (text in code blocks and code spans)
- 'Emphasis' (emphasis)
- 'Strong' (strong emphasis and text in table heading row)
- 'Hyperlink' (hyperlinks)

If style templates are not used, or the corresponding style template is missing, TRichView uses defaults.

Headings

Headings are imported as paragraphs having positive values of OutlineLevel [835].

If StyleTemplates are used [263], 'heading 1' ... 'heading 6' StyleTemplates are applied to heading paragraphs.

If StyleTemplates are not used [263] (or the corresponding StyleTemplate does not exist), TRichView uses defaults for text attributes (you can modify them in the global singleton object RVDefaultLoadProperties [1119]: DefaultProperties[] [1028] property).

Block quotes

All content of block quotes is imported as indented paragraphs.

While the export understands only a single block quote level, the import uses indents to display multilevel lists and block quotes.

If StyleTemplates are used [263], 'Quote' or 'Intense Quote' StyleTemplates are applied to paragraphs. Nested quotes are additionally indented by the left indent specified in the used style template.

If StyleTemplates are not used [263], these paragraphs are indented by DefaultItemProperties [452] .ListMarkerIndentPix [447] and colored by DefaultItemProperties [452].BlockQuoteBackColor [446].

Code blocks

All content of code blocks are imported as paragraphs.

The current version of TRichView cannot export code blocks, it can only import them.

If StyleTemplates are used [263], 'HTML Preformatted' StyleTemplates is applied to paragraphs, 'HTML Code' StyleTemplate is applied to text.

If StyleTemplates are not used [263], these paragraphs are colored by DefaultItemProperties [452] .CodeBlockBackColor [446], and text is formatted using RVDefaultLoadProperties [1119] .DefaultMonoSpacedFontName [1028] font.

Text attributes

If StyleTemplates are not used [263], the following rules are applied when importing text:

- emphasis is imported as italic text
- strong emphasis is imported as bold text
- code spans are formatted using RVDefaultLoadProperties [1119].DefaultMonoSpacedFontName [1028] font
- hyperlinks are blue and underlined (you can modify link appearance in the global singleton object RVDefaultLoadProperties [1119]: DefaultProperties[[1028] *rvftHyperlink*] property).

If StyleTemplates are used [263], the following rules are applied when exporting text:

- emphasis is formatted using 'Emphasis' StyleTemplate
- strong emphasis is formatted using Strong' StyleTemplate
- code spans are formatted using 'HTML Code' StyleTemplate
- hyperlinks are formatted using 'Hyperlink' StyleTemplate.

Since only one StyleTemplate can be applied to text, for text having multiple attributes (such as code spans inside emphasis), only the main StyleTemplate is applied, other attributes are applied as if StyleTemplates are not used. The following priorities of StyleTemplates are used (from highest to lowest): 'Hyperlink', 'HTML Code', 'Strong', 'Emphasis'.

The current version of TRichView cannot export code spans, it can only import them.

Lists

List levels are indented according to DefaultItemProperties [452].ListMarkerIndentPix and ListLeftIndentPix [447] properties.

For complex nested list, you can choose either keeping level nesting (include *rvmdloEnforceListLevels* in LoadOptions [453]) or list type (exclude *rvmdloEnforceListLevels* from LoadOptions [453]).

Tables

To enable table loading, include *rvmdeTables* in Extensions [452].

The first row is imported as a header row. Text in this row is formatted as if it has a strong emphasis.

Table cells have widths specified in DefaultItemProperties [452].TableCellWidthPix [447].

6.1.1.5.6.1 Properties

In TRVMarkdownProperties

- DefaultItemProperties [452]
- Extensions [452]
- ImagesPrefix [453]
- LineWidth [453]
- LoadOptions [453]
- NotesSavedAsFootnotes [455]
- SaveHiddenText [456]
- SaveOptions [456]

Default properties of items used for Markdown loading

```
property DefaultItemProperties: TRVMarkdownDefaultItemProperties[445];
```

(introduced in version 20)

See details in the topic about TRVMarkdownDefaultItemProperties[445].

Lists Markdown extensions supported by TRichView

```
type // defined in RVStyle unit
  TRVMarkdownExtension = (
    rvmdeTables,
    rvmdeFootnotes);
  TRVMarkdownExtensions = set of TRVMarkdownExtension;
property Extensions: TRVMarkdownExtensions;
```

(introduced in version 19)

Value	Extension
rvmdeTables	Tables
rvmdeFootnotes	Footnotes

Tables extension

If support of this extension is activated, TRichView can save and load tables[179] in Markdown format.

Limitation:

- no table properties are saved
- merged cells are saved as empty cells
- only single line cell content is supported (no lists, quotes, headings in cells); however, line breaks can be exported as <br>, if *rvmdsoUseBR* is included in SaveOptions[456]
- center or right alignment is stored for a column, if the first paragraphs of all cells in this column have this alignment
- nested tables are not saved.

If support of this extension is not activated, TRichView saves content of each table cells as normal paragraphs.

Footnotes extension

If support of this extension is activated, TRichView can save footnotes[185], endnotes[184], sidenotes[186] and text boxes[188] in Markdown format as footnotes. See NotesSavedAsFootnotes[455].

If support of this extension is not activated, TRichView saves content of notes directly in text, inside [ ] characters.

Default value:

[*rvmdeTables, rvmdeFootnotes*]

A prefix for file names of images that are saved together with Markdown text.

```
property ImagesPrefix: TRVUnicodeString[1099];
```

(introduced in version 19)

This prefix is used to generate image file names while storing Markdown text.

The following item types are stored as images: pictures[169], hot-pictures[172], bullets[176], hotspots[178].

For example, if ImagesPrefix = 'img', images may be saved as img1.jpg, img2.jpg, etc. (file extensions depend on the image format).

If *rvmdsoOverrideImages* is included in SaveOptions[456], the numbering of images is strictly 1, 2, 3, ...

If it is not included, existing image files are not overridden (for example, if img2.jpg already exists, images are stored as img1.jpg, img3.jpg, ...)

Default value:

'img'

Specifies the maximal line width in Markdown text saved by TRichView.

```
property LineWidth: Integer;
```

(introduced in version 19)

The component wraps text so that every line is at most **LineWidth** characters long (if possible).

In some content (like ATX headings, tables, footnotes) line breaks are not allowed, so TRichView ignores this property for them.

This value must be large enough to include all necessary indents.

Assign **LineWidth** = 0 to turn off line wrapping in Markdown files.

This property has a pure cosmetic effect, Markdown readers ignore line breaks.

Default value:

80

Options for Markdown saving.

```
type // defined in RVStyle unit
  TRVMarkdownLoadOption = (
    rvmdloBasePathLinks,
    rvmdloCollapseWhitespace,
    rvmdloCollapseWhitespaceInCodeSpans,
    rvmdloEnforceListLevels,
    rvmdloKeepLineBreaks);

  TRVMarkdownLoadOptions = set of TRVMarkdownLoadOption;
```

(introduced in version 19)

Option	Meaning
rvmdloBasePathLinks	If included, a base path is added to hyperlink targets (if they are relative paths). For images, a base path is added even if this option is not included (if image paths are relative paths)
rvmdloCollapseWhitespace	If included: replace all runs of whitespace characters (including spaces and tabs) by a single space (according the the Markdown specification). This property is applied to all text except for code spans and code blocks (in code blocks, whitespaces are loaded as they are)
rvmdloCollapseWhitespaceInCodeSpans	If included: replace all runs of whitespace characters (including spaces and tabs) by a single space in code spans (according the the Markdown specification).
rvmdloEnforceListLevels	If included: keeping list nesting levels. If included: keeping list type.
rvmdloKeepLineBreaks	Normally, double spaces or '\' at the end of lines in paragraph produce line breaks. If this option is included, line breaks in Markdown paragraphs produce line breaks as well. This option does not affect code spans (in code spans, line breaks are always replaced by spaces) and code blocks (in code blocks, line breaks always start new paragraphs)

About loading list levels from Markdown

In Markdown, lists have an HTML model: single-level lists that can be nested.

In TRichView (like in Microsoft Word), lists may have multiple levels but cannot be nested.

As a result, the following list cannot be loaded as it is (because it has different type of markers at the second level):

```
* itemA
  * itemAA
* itemB
  1. itemBA
```

If *rvmdloEnforceListLevels* is included, this list is loaded as:

```
• itemA (level 1)
  o itemAA (level 2)
• itemB (level 1)
  o itemBA (level 2)
```

This list can be saved back as:

```
* itemA
  * itemAA
* itemB
  * itemBA
```

If *rvmdloEnforceListLevels* is excluded, this list is loaded like the following list:

```
• itemA (level 1)
  o itemAA (level 2)
• itemB (level 1)
  1. itemBA (level 3)
```

This list can be saved back as:

```
* itemA
  * itemAA
* itemB
  *
    1. itemBA
```

Default value:

[rvmdloBasePathLinks, rvmdloCollapseWhitespace, rvmdloCollapseWhitespaceInCodeSpans, rvmdloEnforceListLevels]

See also

- TRVRTFReaderProperties[457].BasePathLinks[459], TRVHTMLReaderProperties[432].BasePathLinks[433]
- TRVHTMLReaderProperties[432].EnforceListLevels[433]

Lists notes that will be stored in Markdown as footnotes

```
type // defined in RVStyle unit
  TRVNoteType = (rvntFootnote, rvntEndnote, rvntSidenote, rvntTextBox);
  TRVNoteTypes = set of TRVNoteType;
property NotesSavedAsFootnotes: TRVNoteTypes;
```

(introduced in version 19)

Value	Note type
rvntFootnote	Footnote[185]
rvntEndnote	Endnote[184]
rvntSidenote	Sidenote[186]
rvntTextBox	Text box[188]

This property is used only if *rvmdeFootnotes* is included in Extensions [452]. If this extension is activated, notes of the listed types are stored in markdown as footnotes. Otherwise, their text is stored directly in text, inside of [ ].

Only single line note content is supported (no lists, quotes, headings in note text); however, line breaks can be exported as <br>, if *rvmdsoUseBR* is included in SaveOptions [456].

Default value:

[*rvntFootnote, rvntEndnote, rvntSidenote*]

Specifies whether a hidden text [106] is saved to Markdown

```
property SaveHiddenText: Boolean;
```

(introduced in version 19)

Default value:

True

Options for Markdown saving.

```
type // defined in RVStyle unit
  TRVMarkdownSaveOption = (
    rvmdsoUseSetextHeadings,
    rvmdsoCompact,
    rvmdsoListInBlockQuote,
    rvmdsoUseBR,
    rvmdsoOverrideImages,
    rvmdsoUseItemImageFileNames,
    rvmdsoInlineImages,
    );
  TRVMarkdownSaveOptions = set of TRVMarkdownSaveOption;
property SaveOptions: TRVMarkdownSaveOptions;
```

(introduced in version 19, updated in v21)

Option	Meaning
rvmdsoUseSetextHeadings	Using Setext heading (underlined paragraphs) instead of ATX headings (text surrounded by '#' characters) when possible
rvmdsoCompact	Storing compact (but less readable) Markdown
rvmdsoListInBlockQuote	This option is used for paragraphs that have both bullets/numbering and a block-quote style template [103]. If not included, they are saved as block quotes in lists. If included, they are saved as lists in block quotes

Option	Meaning
rvmdsoUseBR	Saving line (or paragraph) breaks as in places where Markdown does not allow line breaks. If this option is not included, they are saved as space characters.
rvmdsoOverrideImages	If set, TRichView can override existing images. If not set, TRichView generates unique names for images.
rvmdsoUseItemImageFileNames	If included, images with defined (non-empty) file names will not be saved, but their file names will be written in Markdown. If this option is included, OnSaveImage2 [410] event has initial value of the Location parameter equal to the image file name. File names for items are defined via *rvespImageFileName* string property [1072].
rvmdsoInlineImages	If included, images are stored directly in Markdown text (in base64 encoding). Otherwise images are saved in external files.

Default value:

- for TRichView [216], TRichViewEdit [469], TDBRichView [604]: [*rvmdsoUseSetextHeadings, rvmdsoUseItemImageFileNames*]
- for TDBRichViewEdit [616], TcxTRichViewEdit.Properties: [*rvmdsoUseSetextHeadings, rvmdsoUseItemImageFileNames, rvmdsoInlineImages*]

See also

- TRVHTMLSaveProperties [436].ImageOptions [442] (has similar options for saving images in HTML)

6.1.1.5.7 TRVRTFReaderProperties

This is a type of TRichView [216].RTFReadProperties [253]. It contains properties for controlling RTF and DocX import in TRichView and TRichViewEdit.

Do not create objects of this class yourself, use this property instead.

Unit [VCL/FMX] RVRTFProps / fmxRVRTFProps.

Syntax

```
TRVRTFReaderProperties = class (TPersistent)
```

Properties

The most important properties are:

- ParaStyleMode [464];
- TextStyleMode [466].

The following properties allow to ignore some part of RTF content:

- IgnoreBookmarks [463];
- IgnoreFields [463];
- IgnoreNotes [463];
- IgnorePictures [463];
- IgnoreSequences [463];
- IgnoreTables [464];
- SkipHiddenText [466];
- ReadDocParameters [465].

Reading Hyperlinks

By default, hyperlinks targets are loaded in items tags [96]. Use TCustomRichView.OnReadHyperlink [395] to customize loading.

Reading Images

The following types of RTF pictures are supported (if IgnorePictures [463] = *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 [462];
- 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 [1125].RegisterPngGraphic);
- external images, loaded from files (TCustomRichView.OnImportPicture [386] event occurs).

DocX may contain images of other types (such as Gif).

6.1.1.5.7.1 Properties

In TRVRTFReaderProperties

- AutoHideTableGridLines [459]
- BasePathLinks [459]
- CanReuseNumberedLists [459]
- CharsetForUnicode [460]
- ConvertHighlight [460]
- ConvertSymbolFonts [461]
- DefCharsetCodePage [461]
- DocXErrorCode [462]
- ExtractMetafileBitmaps [462]
- IgnoreBookmarks [463]
- IgnoreFields [463]
- IgnoreNotes [463]
- IgnorePictures [463]
- IgnoreSequences [463]
- IgnoreTables [464]

- LineBreaksAsParagraphs 464
- ParaStyleMode 464
- ParaStyleNo 465
- ReadDocParameters 465
- ReadNoteNumberingType 465
- RTFErrorCode 466
- SkipHiddenText 466
- TextStyleMode 466
- TextStyleNo 467
- UseCharsetForUnicode 468
- UseHypertextStyles 468
- UseSingleCellPadding 468

Hides grid lines (dotted lines in place of hidden borders) for tables imported from RTF and DocX.

```
property AutoHideTableGridLines: Boolean;
```

(introduced in version 1.6)

If this property is set to *True*, RichView includes *rvtoHideGridLines* in Options 916 of tables read from RTF and DocX.

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 247.

Default value:

False

Specifies whether to add base path (path to RTF/DocX 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/DocX loading.

For example, when loading "c:\Docs\MyFile.rtf" containing link to "MyFile2.rtf":

- if this property is *False*, TCustomRichView.OnReadHyperlink 395 event occurs with Target='MyFile2.rtf' (or, if the event is not assigned, this value is stored in them item tag 96);
- 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).

See also:

- TRichView.HTMLReadProperties 243.BasePathLinks 433
- rvmdloBasePathLinks option in TRichView.MarkdownProperties 245.LoadOptions 453

Default value:

True

Specifies whether numbered paragraph lists in inserted RTF/DocX content can continue numbering in existing lists.

```
property CanReuseNumberedLists: Boolean;
```

(introduced in version 11)

If **CanReuseNumberedLists** = *True*, methods for RTF/DocX insertion can merge existing numbered paragraph lists with lists in inserted content (if they have identical properties).

If this is undesirable, assign **CanReuseNumberedLists** = *False*, and inserted lists will be independent.

Default value:

True

Specifies a character set [809] for Unicode text read from RTF and DocX.

```
property CharsetForUnicode: TRVFontCharset[1075];
```

(introduced in version 11)

This property is used if UseCharsetForUnicode [468] =*True* and TextStyleMode [466] <>*rvrsUseSpecified*.

This character set is applied to Unicode text read from RTF and DocX (in DocX, all text is Unicode).

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.

This property is not used for DocX import.

Default value:

rtfhlColorTable

Specifies whether text of symbol fonts is converted to Unicode text.

```
property ConvertSymbolFonts: Boolean;
```

(introduced in version 20)

If a font has Charset = SYMBOL_CHARSET, it means that this font contains a set of glyphs specific for this font. Text formatted with this font can be displayed properly only if this font is installed. Even if these fonts are common on Microsoft Windows computers, they can be missing on other platforms.

Because of this problem, it is desirable to use fonts that follow the Unicode standard. Text formatted with Unicode fonts can be displayed even if this font or some glyphs in it are not available (using a font substitution).

If **ConvertSymbolFonts** = *False*, text of all symbol fonts is loaded as it is.

If **ConvertSymbolFonts** = *True*, characters of widely used symbol fonts are converted to the most similar Unicode characters.

Conversion of the following fonts is supported:

- Symbol
- Wingdings
- Wingdings 2
- Wingdings 3
- Webdings

This converted text will be formatted with the font specified in RVDefaultLoadProperties[1119].DefaultUnicodeSymbolFontName[1028]. The default value for this font name depends on the platform.

Default value:

- VCL and LCL: *False*
- FireMonkey: *True*

A code page used to convert text of *DEFAULT_CHARSET*.

```
property DefCharsetCodePage: TRVCodePage[1062];
```

(introduced in version 24)

Typically, RTF files should not contain text with the charset *DEFAULT_CHARSET*. The text should be represented either in Unicode encoding or as characters of a specific charset.

However, some programs that create RTF files may specify *DEFAULT_CHARSET* for the text. As a result, the characters read from the file will depend on the default language set in the operating system.

This property allows you to override the code page used for text with *DEFAULT_CHARSET*. For example, you can set it to 1252 to use characters from Western alphabets.

This property is only for RTF files, because DocX files contain only Unicode text.

Default value:

0 *(CP_ACP)*

Contains error code for the last DocX reading operation.

```
ErrorCode: TRVRTFErrorCode;
```

```
type // defined in RVDocXReader unit.
  TRVDocXErrorCode = (
    rvdocxreOK,           // No error
    rvdocxreUnpack        // File unpacking error (maybe not a DocX file)
    rvdocxreStyleSheet    // Error while parsing a style sheet
    rvdocxreDocument,     // Error while parsing the main document
    rvdocxreAssertion,    // Assertion failure (internal parser error)
    rvdocxreException,    // Exception (internal parser error)
    rvdocxreFileOpenError // Parser cannot open input file
    );
```

See also:

- RTFErrorCode 466

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 and DocX.

```
property IgnoreBookmarks: Boolean;
```

(introduced in version 10)

Bookmarks starts are read as *checkpoints* [92]. Bookmark ends cannot be read.

Default value:

False

Disallows loading "page number" [190] and "page count" [192] fields from RTF and DocX.

```
property IgnoreFields: Boolean;
```

(introduced in version 15)

If *True*, fields are loaded as a plain text.

Default value:

False

Disallows loading footnotes [185], endnotes [184] and references to them [190] from RTF and DocX.

```
property IgnoreNotes: Boolean;
```

(introduced in version 10)

If loading of notes is allowed, their properties are also loaded as well, specifically:

- Style [260].FootnotePageReset [749];
- Style [260].FootnoteNumbering and EndnoteNumbering [746] (if ReadNoteNumberingType [465] = *True*)

Default value:

False

Disallows loading pictures from RTF and DocX.

```
property IgnorePictures: Boolean;
```

Set it to *True* to import RTF and DocX 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 [462] property);
- Enhanced Metafiles, loaded as TMetafile,
- JPEG (D3+), loaded as TJpegImage,
- PNG, see RVGraphicHandler [1125].RegisterPngGraphic.

Default value:

False

Disallows loading numbering sequences [183] from RTF and DocX.

```
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 and DocX.

```
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 and DocX.

```
property ParaStyleMode: TRVReaderStyleMode[1086];
```

This property defines how RTF/DocX paragraph attributes will be converted to TRichView.Style[260].ParaStyles[757].

Mode	Meaning
rvrsUseSpecified	All paragraph formatting in RTF/DocX will be ignored, all paragraphs will have style specified in ParaStyleNo[465] property.
rvrsUseClosest	RTF/DocX formatting will be mapped to the the most similar existing paragraph style[757]
rvrsAddIfNeeded	This mode provides the most closer look to the original RTF/DocX. TRichView will try to use existing styles as much as possible, but if a style with the desired formatting does not exist, TRichView will add a new one to the end of the collection of paragraph styles[757] and use it.

List markers (bullets and numbering) are imported only if ParaStyleMode=*rvrsAddIfNeeded*.

If the both TextStyleMode[466]=**ParaStyleMode**=*rvrsAddIfNeeded*, and TRichView.UseStyleTemplates[263]=*True*, and StyleTemplateInsertMode[260] <>*rvstimIgnoreSourceStyleTemplates,* the RTF style sheet will be read in TRichView.Style[260]

.StyleTemplates[762]. For RTF/DocX loading, the RTF/DocX style sheet replaces existing style templates. For RTF/DocX insertion, the RTF/DocX style sheet is merged into the existing style templates. In this case, RTF/DocX paragraph properties are adjusted according to TRichView.StyleTemplateInsertMode[260].

Default value:

- *rvrsUseClosest* for VCL/LCL components created at runtime.
- *rvrsAddIfNeeded* for FireMonkey components created at runtime.
- For components placed on form at designtime, initial value of this property is defined in the component editor[224] 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[465];
- TextStyleMode[466].

Default paragraph style for RTF and DocX loading.

```
property ParaStyleNo: Integer
```

This property is used if ParaStyleMode[464]=*rvrsUseSpecified*.

It is an index in the collection of paragraph styles[757].

Default value:

0

See also:

- ParaStyleMode[464];
- TextStyleNo[467].

Set to True to allow reading DocParameters[237] from RTF and DocX.

```
property ReadDocParameters: Boolean;
```

(introduced in version 10)

Default value:

False

Set to True to allow reading Style[260].FootnoteNumbering and EndnoteNumbering[746] from RTF and DocX.

```
property ReadNoteNumberingType: Boolean;
```

(introduced in version 22)

This property is used only if IgnoreNotes[463] = *False*.

Default value:

False

Contains error code for the last RTF reading operation.

```
RTFErrorCode: 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)
    );
```

See also:

- DocXErrorCode [462]

This property controls how hidden text is loaded from RTF and DocX

```
property SkipHiddenText: Boolean;
```

(introduced in version 1.6)

If *True*, hidden content is not added to TRichView.

If *False*, hidden content [106] is added.

Default value:

True

See also:

- TRVHtmlReaderProperties [432].SkipHiddenText [435]

Defines a mode for extraction of character formatting from RTF and DocX

```
property TextStyleMode: TRVReaderStyleMode[1086];
```

This property defines how RTF/DocX text attributes will be converted to TRichView.Style [260].TextStyles [764].

Mode	Meaning
rvrsUseSpecified	All character formatting in RTF/DocX will be ignored, all text will have style specified in TextStyleNo [467] property.

rvrsUseClosest	RTF/DocX formatting will be mapped to the the most similar existing text style[764].
rvrsAddIfNeeded	This mode provides the most closer look to the original RTF/DocX. TRichView will try to use existing styles as much as possible, but if a style with the desired formatting does not exist, TRichView will add a new one to the end of the collection of text styles[764] and use it.

If the both **TextStyMode**=ParaStyleMode[464]=*rvrsAddIfNeeded*, and TRichView.UseStyleTemplates[263]=*True*, and StyleTemplateInsertMode[260] <>*rvstimIgnoreSourceStyleTemplates,* the RTF/DocX style sheet will be read in TRichView.Style[260].StyleTemplates[762]. For RTF/DocX loading, the RTF/DocX style sheet replaces existing style templates. For RTF/DocX insertion, the RTF/DocX style sheet is merged into the existing style templates. In this case, RTF/DocX character properties are adjusted according to TRichView.StyleTemplateInsertMode[260].

Default value:

- *rvrsUseClosest* for VCL/LCL components created at runtime.
- *rvrsAddIfNeeded* for FireMonkey components created at runtime.
- For components placed on form at designtime, initial value of this property is defined in the component editor[224] 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[467];
- ParaStyleMode[464].

Default text style for RTF and DocX loading.

```
property TextStyleNo: Integer
```

This property is used if TextStyleMode[466]=*rvrsUseSpecified*.

It is an index in the collection of text styles[764].

Default value:

0

See also:

- TextStyleMode[466];
- ParaStyleNo[465].

Obsolete property, it does nothing

```
property UnicodeMode: TRVReaderUnicode;

type
  TRVReaderUnicode =
    (rvruMixed, rvruNoUnicode, rvruOnlyUnicode);
```

Specifies whether CharsetForUnicode[460] property is used as a character set for Unicode text read from RTF and DocX.

```
property UseCharsetForUnicode: Boolean;
```

Default value:

False

```
property UseHypertextStyles: Boolean
```

Set this property to *True* to allow using existing hypertext text styles for non-hypertext text from RTF (in TextStyleMode[466] = *rvrsUseClosest* or *rvrsAddIfNeeded*).

Note: it's not recommended to change value of this property.

Default value:

False

See also:

- Hypertext in TRichView[97].

Set it to *True* if you want to use the same values for CellHPadding[911] and CellVPadding[913] 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[129] files containing tables with different cell padding).

Default value:

False

6.1.1.6 Examples

TRichView Examples

- How to load a UTF-8-encoded text file[468]

6.1.1.6.1 Loading UTF-8 files

This example shows how to load Unicode UTF-8 file:

```
procedure LoadUTF8(rv: TCustomRichView[216]; const FileName: String;
  StyleNo, ParaNo: Integer);
var Stream: TFileStream;
    s: TRVRawByteString[1085];
    ws: TRVUnicodeString[1099];
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(281)(ws, StyleNo, ParaNo, ParaNo, False);
end;
```

Call:

```
LoadUTF8(RichViewEdit1, 'test.txt', 0, 0);
RichViewEdit1.Format(295);
```

See also:

- Unicode in TRichView(135).

6.1.2 TRichViewEdit

TRichViewEdit is a control for editing documents with pictures, tables, Delphi controls and hyperlinks.

Unit [VCL/FMX]: RVEdit / fmxRVEdit.

Syntax

```
TCustomRichViewEdit = class(TCustomRichView(216))
TRichViewEdit = class(TCustomRichViewEdit)
```

RichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style(260) property.

The most important property settings can be done in the component editor(224) 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(379) event;
- all features related to hypertext ids:OnRVMouseMove(403), OnJump(390), GetJumpPointLocation(312) work only when user presses and holds Ctrl key, if *rvtoCtrlJumps* in EditorOptions(483) (or if ReadOnly(488)=*True*).

Hierarchy

Hierarchy (VCL and LCL)

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRVControl
TRVScroller(869)
TCustomRichView(216)
TCustomRichViewEdit

Hierarchy (FireMonkey)

TObject

TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBox
TCustomRVControl
TRVScroller 869
TCustomRichView 216
TCustomRichViewEdit

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

- Properties 470
- Events 470
- General methods 471
- Inserting items at the position of caret 471
- Clipboard operations 471
- Advanced methods for modifying items 472
- Information about item at the position of caret 472
- Modifying item at the position of caret 473
- Style templates 473
- Undo/Redo 473
- Unicode 474
- Live spelling check 474
- "Smart Popups" 474
- Related to tables and special items 474
- Working with *checkpoints* 475
- LiveBindings and DB 475

Properties

- CurItemNo 480 – index of item at the position of caret, OffsetInCurItem 488 – offset of caret in this item;
- CurItemStyle 481 – style (or type) of the item at the position of caret;
- CurParaStyleNo 481 – current paragraph style index;
- CurTextStyleNo 482 – current text style index, see also ActualCurTextStyleNo 480;
- EditorOptions 483 – options for editing;
- ReadOnly 488 disables/enables editing;
- Modified 487 – all editing methods set it to *True*.
- CustomCaretInterval 483 – a blinking interval for a custom caret.
- AcceptDragDropFormats, AcceptPasteFormats 478 – lists of formats that the editor can accept.
- DefaultPictureVAlign 483 – alignment for pasted and dropped pictures.
- ForceFieldHighlight 486 affects highlighting of label items 181.
- KeyboardType 487 [FMX] – the type of the virtual keyboard

See also properties for undo/redo.

Events

- OnChange 582 occurs when the document is changed;
- OnCheckStickingItems 583 allows to add additional text protection;
- OnCurParaStyleChanged 583 occurs when the index of the current paragraph style is changed;
- OnCurTextStyleChanged 584 occurs when the index of the current text style is changed;
- OnPaste 594 allows pasting data from the Clipboard in custom formats;

- OnStyleConversion[595] occurs while executing ApplyStyleConversion[501];
- OnParaStyleConversion[592] occurs while executing ApplyParaStyleConversion[499];
- OnCaretGetOut[581] occurs when the user moves the caret "outside" the editor;
- OnCaretMove[582] occurs when the caret is moved;
- OnItemTextEdit[588] occurs when a text item[167]'s text is changed as a result of editing operation;
- OnSmartPopupClick[595] occurs when user clicks a "smart popup"[474] button.
- OnDrawCurrentLine[584] allows to highlight the current line
- OnMeasureCustomCaret[588] and OnDrawCustomCaret[585] allow drawing your own caret instead of the system one.
- OnOleDragEnter[589], OnOleDragOver[590], OnOleDragLeave[589], OnOleDrop[591], OnBeforeOleDrop and OnAferOleDrop[580] occur on OLE drag&drop.

General Methods

- ApplyParaStyle[498], ApplyTextStyle[502] apply paragraph and text style to the selection;
- ApplyStyleConversion[501] applies custom style conversion to the selected text;
- ApplyParaStyleConversion[499] applies custom paragraph style conversion to the selected paragraphs;
- Change[508] generates OnChange[582] event;
- BeginUpdate[506], EndUpdate[512] disable and enable redrawing;
- SearchText, -A, -W[553] search for substring;
- MoveCaret[542] moves the caret in the specified direction.

Inserting Items at the position of Caret (Insert*** methods)

- InsertText, -A, -W[540], InsertStringTag, InsertStringATag, InsertStringWTag[539] insert text;
- InsertBreak[523] inserts *break* (horizontal line);
- InsertBullet[524] inserts *bullet*;
- InsertControl[526] inserts Delphi control;
- InsertHotspot[528] inserts *hotspot*;
- InsertPicture[534] inserts picture;
- InsertHotPicture[528] inserts picture-hypertext link;
- InsertTab[540] inserts tabulator;
- InsertRVFFromFileEd[537], InsertRVFFromStreamEd[538] insert RVF (RichView Format) from a file or a stream;
- InsertHTMLFromFileEd[529], InsertHTMLFromStreamEd[530] insert HTML from a file or a stream;
- InsertRTFFromFileEd[536], InsertRTFFromStreamEd[536] insert RTF (Rich Text Format) from a file or a stream;
- InsertTextFromFile[541],InsertOEMTextFromFile[533],InsertTextFromFileW[541] insert text from a file;
- nsertMarkdownFromFileEd[532],InsertMarkdownFromStreamEd[532] insert Markdown from a file or a stream;
- InsertItem[531] – general method for inserting items, inserts one item (usually used for tables).

Clipboard Operations

Main methods and events:

- CanPaste[507] – "can something be pasted from the Clipboard?";
- CanPasteRVF[507] – "can RVF (RichView Format) be pasted from the Clipboard?";
- CanPasteHTML[507] – "can HTML be pasted from the Clipboard?";
- CanPasteRTF[507] – "can RTF (Rich Text Format) be pasted from the Clipboard?";

- CutDef[511] cuts selection to the Clipboard;
- Paste[543] pastes from the Clipboard;
- PasteBitmap[544] pastes bitmap from the Clipboard;
- PasteMetafile[546] pastes metafile from the Clipboard;
- PasteGraphicFile[544] pastes metafile from the Clipboard;
- PasteRVF[547] pastes RVF (RichView Format) from the Clipboard;
- PasteHTML[545] pastes RTF (Rich Text Format) from the Clipboard;
- PasteRTF[547] pastes RTF (Rich Text Format) from the Clipboard;
- PasteText[548] pastes Unicode text from the Clipboard;
- PasteMarkdown[546] pastes Unicode text from the Clipboard as Markdown;
- event OnPaste[594] allows to paste data from the Clipboard in custom formats.

Also:

- DeleteSelection[511] deletes selection;
- SelectCurrentWord[555] selects word at the position of caret;
- SelectCurrentLine[554] selects the line containing caret.

Advanced Methods for Modifying Items (Set***InfoEd methods)

Main methods:

- SetBreakInfoEd[557] – for changing *break*;
- SetBulletInfoEd[558] – for changing *bullet*;
- SetControlInfoEd[560] – for changing inserted control;
- SetHotspotInfoEd[570] – for changing *hotspot*;
- SetPictureInfoEd[576] – for changing picture or *hot picture*.

Also:

- SetItemExtraIntProperty[Ex]Ed[572], SetItemExtraStrProperty[Ex]Ed[573] – for changing additional properties of items;
- SetItemTagEd[574] – for changing *tag*;
- SetItemVAlignEd[575] – for changing item position relative to the line;
- SetItemTextEd[574] changes text (name) of item;
- SetCheckpointInfoEd[560] – for changing *checkpoint*;
- RemoveCheckpointEd[550] – for deleting *checkpoint*;
- ResizeControl[552] – for resizing inserted control;
- AdjustControlPlacement[497] returns control to the proper position;
- AdjustControlPlacement2[497] 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[480],...) but work also in inplace editors of tables[196].

Main methods:

- GetCurrentTextInfo[523] – about text item;
- GetCurrentBreakInfo[512] – about *break*;
- GetCurrentBulletInfo[513] – about *bullet*;
- GetCurrentControlInfo[515] – about inserted control;
- GetCurrentHotspotInfo[515] – about *hotspot*;
- GetCurrentPictureInfo[522] – about picture or *hot picture*;

- GetCurrentItem[517] – as an object about any item (usually for tables[196]);
- GetCurrentItemEx[517].

Also:

- GetCurrentItemExtraIntProperty[Ex][518], GetCurrentItemExtraStrProperty[Ex][519] return value of additional item property;
- GetCurrentTag[522] returns item's *tag*;
- GetCurrentItemVAlign[520] returns item position relative to the line;
- GetCurrentCheckpoint[514] returns item's *checkpoint*;
- GetCurrentItemText[519] returns text/name of item;
- GetCurrentLineCol[521] returns line and column at the caret position.

Modifying Item at Position of Caret (SetCurrent***Info methods)

These methods are equivalent to Set***InfoEd(CurItemNo[480],...) but work also in inplace editors of tables[196].

Main methods:

- SetCurrentBreakInfo[561] modifies *break*;
- SetCurrentBulletInfo[562] modifies *bullet*;
- SetCurrentHotspotInfo[565] modifies *hotspot*;
- SetCurrentControlInfo[564] modifies inserted control;
- SetCurrentPictureInfo[569] modifies picture or *hot picture*.

Also:

- SetCurrentItemExtraIntProperty[Ex][566],
- SetCurrentItemExtraStrProperty[Ex][573] change value of additional item property;
- SetCurrentTag[570] modifies item's *tag*;
- SetCurrentItemVAlign[569] changes item position relative to the line;
- SetCurrentItemText[568] changes text (name) of item;
- SetCurrentCheckpointInfo[563] modifies *checkpoint*;
- RemoveCurrentCheckpoint[551] deletes *checkpoint*;
- ResizeCurrentControl[552] returns control to the proper position in document;

- RemoveCurrentPageBreak[551] removes "explicit page break before item" flag;
- InsertPageBreak[534] sets "explicit page break before item" flag; can split current item into two parts.

Style templates

Methods:

- ApplyStyleTemplate[501], ApplyTextStyleTemplate[502], ApplyParaStyleTemplate[500] apply the specified style template to the selection;
- ChangeStyleTemplates[508] allows changing the collection of style templates as an editing operation.

Undo/Redo

Main:

- property UndoLimit[490] sets capacity of undo buffer;
- UndoAction[577] returns which operation will be undone next;

- Undo[577] undoes the last operation;
- UndoName[578] returns name of custom undo operation which will be undone next;
- RedoAction[549] returns which operation will be redone next;
- Redo[549] redoes the last undone operation;
- RedoName[550] returns name of custom redo operation which will be redone next;
- ClearUndo[509] clears undo and redo buffers.

Undo grouping:

- BeginUndoGroup[505],
- BeginUndoCustomGroup[504],
- SetUndoGroupMode[577].

Alternative undo grouping:

- BeginUndoGroup2, EndUndoGroup2[505];
- BeginUndoCustomGroup2, EndUndoCustomGroup2[504].

Unicode

- PasteTextW[548] pastes Unicode text from the Clipboard;
- InsertTextW[540] inserts Unicode text;
- InsertTextFromFileW[541] inserts Unicode text from file.

Live Spelling Check

- GetCurrentMisspelling[521] returns misspelled word at the caret position;
- [FMX] AddSpellingMenuItems[497] adds spelling check suggestions to a popup menu;
- property LiveSpellingMode[487] controls when live spelling thread starts;
- [FMX] event OnGetSpellingSuggestions[382] requests a list of possible corrections for a misspelled word.

"Smart Popups"

"Smart popups" are buttons that can be used to set properties for the current (at the position of caret) item.

Properties:

- SmartPopupProperties[489] – popup button properties;
- SmartPopupVisible[489] – popup button properties.

Events:

- OnSmartPopupClick[595] occurs when user clicks "smart popup" button.

Related to Tables[901] and Special Items

- InsertItem[531] – general method for inserting items, inserts one item (usually used for tables[179], labels[181], sequences[183], footnotes[185], endnotes[184], references to notes[190]).
- GetCurrentItem[517] returns item at the position of caret as an object (usually used for tables[179], labels[181], sequences[183], footnotes[185], endnotes[184], references to notes[190]);
- BeginItemModify[503], EndItemModify[511] are used to reformat editor after operations on item (usually on tables);
- CanChange[506] checks possibility of performing editing operation (usually it is called before making operations on table).

Working with *Checkpoints* 92

Editing-style analogs of methods of TRichView 216:

- SetCheckpointInfoEd 560 – adds *checkpoint* for the specified item;
- RemoveCheckpointEd 550 – deletes *checkpoint* for the specified item.

Methods working with the item at the position of caret:

- GetCurrentCheckpoint 514 – returns *checkpoint* associated with the item at the caret position;
- SetCurrentCheckpointInfo 563 – adds/modifies *checkpoint* for the item at the caret position;
- RemoveCurrentCheckpoint 551 – deletes *checkpoint* for the item at the caret position.

Methods working with the checkpoint exactly at the position of caret:

- InsertCheckpoint 525 – inserts *checkpoint* in the caret position;
- GetCheckpointAtCaret 512 – returns the *checkpoint* exactly at the caret position;
- RemoveCheckpointAtCaret 550 – deletes the *checkpoint* exactly at the caret position.

LiveBindings and Databases

Events:

- OnSaveCustomFormat 595 allows saving data in DB field in custom formats

6.1.2.1 Properties

Derived from TCustomRichViewEdit

- AcceptDragDropFormats 478
- AcceptPasteFormats 478
- ActualCurTextStyleNo 480
- CurItemNo 480
- CurItemStyle 481
- CurParaStyleNo 481
- CurTextStyleNo 482
- CustomCaretInterval 483
- DefaultPictureVAlign 483
- EditorOptions 483
- ForceFieldHighlight 486
- KeyboardType 487 [FMX]
- LiveSpellingMode 487
- Modified 487
- OffsetInCurItem 488
- ReadOnly 488
- SmartPopupProperties 489
- SmartPopupVisible 489
- TopLevelEditor 489
- UndoLimit 490

Derived from TCustomRichView 216

- AnimationMode 228
- BackgroundPicture 229

- BackgroundProperties 229
- BiDiMode 230
- BottomMargin 231
- ClearLeft 232
- ClearRight 232
- Color 233
- CPEventKind 233
- Cursor 234
- DarkMode 234
- Delimiters 235
- DocObjects 235
- DocParameters 237
- DocProperties 238
- Document 238
- DocumentHeight 239
- DocumentPixelsPerInch 239 [VCL,LCL]
- DoInPaletteMode 241
- FirstItemVisible 243
- FirstJumpNo 243
- HTMLReadProperties 243
- HTMLSaveProperties 244
- ItemCount 244
- LastItemVisible 244
- LeftMargin 245
- MarkdownProperties 245
- MaxLength 246
- MaxTextWidth 246
- MinTextWidth 246
- NoteText 247
- Options 247
- PageBreaksBeforeItems 250
- RightMargin 251
- RTFOptions 252
- RTFReadProperties 253
- RVData 254
- RVFOptions 254
- RVFParaStylesReadMode 258
- RVFTextStylesReadMode 258
- RVFWarnings 258
- SingleClick 260 (deprecated)
- SelectionHandlesVisible 260 (D2010+)
- **Style** 260
- StyleTemplateInsertMode 260
- TabNavigation 262
- TopMargin 262
- UseFMXThemes 264 [FMX]
- UseStyleTemplates 263
- UseVCLThemes 264 [VCL]

- VAlign [264]
- VSmallStep [264] [VLC, LCL]
- WordWrap [265]
- ZoomPercent [265] [FMX]

Derived from TRVScroller [869]

- BorderStyle [871] [VCL]
- HScrollMax [871]
- HScrollPos [872]
- HScrollVisible [872] [VCL,LCL]
- InplaceEditor [872]
- NoVScroll [873]
- Tracking [873]
- UseXPThemes [873] [VCL,LCL]
- VScrollMax [874]
- VScrollPos [874]
- VScrollVisible [874] [VCL,LCL]
- WheelStep [875] [VCL,LCL]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- Align
- Anchors
- AutoHide [FMX]
- Constraints
- Ctl3D [VCL]
- DisableMouseWheel [FMX]
- DragMode
- EnableDragHighlight [FMX]
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- KeyboardType [FMX]
- Left [VCL, LCL]
- Locked [FMX]
- Margins [FMX]
- Name
- Opacity [FMX]
- Padding [FMX]
- ParentCtl3D [VCL]
- ParentShowHint
- PopupMenu
- Position [FMX]
- RotationAngle [FMX]

- RotationCenter [FMX]
- Scale [FMX]
- ShowHint
- ShowScrollBars [FMX]
- ShowSizeGrip [FMX]
- Size [FMX]
- StyleLookup [FMX]
- TabOrder
- TabStop
- Touch (D2010+) [VCL,FMX]
- TouchTargetExpansion [FMX]
- Tag
- Top [VCL, LCL]
- Visible
- Width

6.1.2.1.1 TRichViewEdit.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 (129)
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 (395) event occurs with DocFormat=*rvlfURL*. If this event

Value	D&D Format	Meaning
		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 [586] event (only drag&drop).
rvddHTML	'HTML Format'	HTML

Example (processing OnReadHyperlink [395] with *rvddURL* support)

```
// OnReadHyperlink
procedure TMyForm.MyRichViewEditReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[1099]; DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag[1096];
  var ItemName: TRVUnicodeString[1099]);
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[821];
begin
  Result := rve.CurTextStyleNo[482];
  TextStyle := TFontInfo.Create(nil);
  try
    TextStyle.Assign[820](rve.Style[260].TextStyles[764][Result]);
    TextStyle.Jump[823] := True;
    TextStyle.Color[810] := clBlue;
    TextStyle.Style[817] := TextStyle.Style+[fsUnderline];
    Result := rve.Style.FindTextStyle[771](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[543];
- CanPaste[507];
- Drag&Drop[123].

6.1.2.1.2 TRichViewEdit.ActualCurTextStyleNo

Index of the current text style in the collection of text styles. Can return rvsDefStyle[1127].

```
property ActualCurTextStyleNo: Integer;
```

(introduced in v1.8)

This is an index in the collection Style[260].TextStyles[764].

This property is the same as CurTextStyleNo[482], but:

- read-only;
- does not process *rvsDefStyle*[1127], returns it as it is.

6.1.2.1.3 TRichViewEdit.CheckSpelling

Allows using a spelling check service provided by the current platform.

```
property CheckSpelling: Boolean;
```

In FireMonkey, there are two ways to implement a live spelling check[138]:

- using OnSpellingCheck[416] and OnGetSpellingSuggestions[382] events
- using a spelling checking service (IFMXSpellCheckerService) provided by several platforms (for example, macOS).

Assign *True* to this property to allow using a platform spelling check service.

Assigning *True* to this property does not start checking immediately, it starts according to LiveSpellingMode[487]. Assigning *False* to this property stops live spelling checking immediately.

Default value:

False

See also:

- Live spelling check in TRichView[138].

6.1.2.1.4 TRichViewEdit.CurItemNo

Index of item at the position of caret

```
property CurItemNo: Integer;
```

This value is in range from 0 to ItemCount[244]-1.

If the caret is inside table cell, this property returns index of the table in the main document. Items of subdocuments (table cells[951]) 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[489].CurItemNo.

This property must be accessed only when the document is formatted.

When called for an empty editor (i.e. ItemCount[244]=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[1047] unit instead.

See also properties:

- OffsetInCurItem[488];
- TopLevelEditor[489];
- CurItemStyle[481];
- CurTextStyleNo[482];
- CurParaStyleNo[481].

See also events:

- OnCaretMove[582].

See also:

- Inserting at the position of the caret[119];
- How to: move the caret to the beginning or to the end of document in TRichViewEdit[1133].

6.1.2.1.5 TRichViewEdit.CurItemStyle

Style (i.e. type) of item at the position of caret

```
property CurItemStyle: Integer;
```

The same as TopLevelEditor[489].GetItemStyle[310](TopLevelEditor[489].CurItemNo[480]).

If the current item is a text item, this function returns zero or positive value: index in the collection of text styles (Style[260].TextStyles[764]).

Otherwise, this function returns a type of this item: see rvsXXX constants[1127].

This property must be accessed only when the document is formatted.

See also methods:

- GetItemStyle[310].

See also properties:

- CurItemNo[480];
- CurTextStyleNo[482].

See also:

- Item types[90].

6.1.2.1.6 TRichViewEdit.CurParaStyleNo

Index of the current paragraph style in the collection of paragraph styles.

```
property CurParaStyleNo: Integer;
```

This is an index in the collection Style[260].ParaStyles[757].

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[583] event occurs.

Usually you do not need not to assign value to this property directly, use ApplyParaStyle [498] or ApplyParaStyleConversion [499] methods instead.

This property must be accessed only when the document is formatted.

See also methods:

- ApplyParaStyle [498];
- ApplyParaStyleConversion [499].

See also properties:

- CurTextStyleNo [482].

See also events:

- OnCurParaStyleChanged [583].

See also:

- Working with paragraphs [100].

6.1.2.1.7 TRichViewEdit.CurTextStyleNo

Index of the current text style in the collection of styles.

```
property CurTextStyleNo: Integer;
```

This is an index in the collection Style [260].TextStyles [764].

This text style will be used when inserting new text in the document at the position of caret (both by user keyboard input and by methods, such as PasteText [548] or InsertText [540]).

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 [584] event occurs.

Value of this property is always positive. To get style (i.e. type) of item at the caret position, use CurItemStyle [481] property.

The editor never sets this property to index of text style having *rvprDoNotAutoSwitch* in Protection [815]. You can do it yourself to insert a special symbol (by keyboard input, or InsertText [540]), but after that the component automatically switches this property back to the last used style without *rvprDoNotAutoSwitch* in Protection [815].

Usually you do not need to assign value to this property directly, use ApplyTextStyle [502] or ApplyStyleConversion [501] instead.

This property must be accessed only when the document is formatted.

This property never returns *rvsDefStyle* [1127] 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* [1127] style, use ActualCurTextStyleNo [480] property.

See also methods:

- ApplyTextStyle [502];
- ApplyStyleConversion [501].

See also properties:

- CurItemStyle [481];
- CurParaStyleNo [481];

- ActualCurTextStyleNo[480].

See also events:

- OnCurTextStyleChanged[584].

6.1.2.1.8 TRichViewEdit.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[295] method (and is deactivated when you call Clear[286]).

It's not recommended to use the custom caret feature together with image animation[124].

Default value:

0 (the system caret is used)

See also events:

- OnMeasureCustomCaret[588];
- OnDrawCustomCaret[585].

6.1.2.1.9 TRichViewEdit.DefaultPictureVAlign

Specifies the default alignment for pasted images, and for images inserted as a result of drag and drop operations.

```
property DefaultPictureVAlign: TRVVAlign[1100];
```

(introduced in version 14)

The following methods use this property:

- PasteBitmap[544]
- PasteMetafile[546]
- PasteGraphicFile[544]
- Paste[543] (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

6.1.2.1.10 TRichViewEdit.EditorOptions

Additional options affecting editing

```
type
  TRVEditorOption = (rvoClearTagOnStyleApp, rvoCtrlJumps,
    rvoDoNotWantReturns, rvoWantTabs,
    rvoAutoSwitchLang, rvoHideReadOnlyCaret,
    rvoDoNotWantShiftReturns, rvoNoImageResize,
    rvoNoCaretHighlightJumps, rvoNoReadOnlyJumps,
    rvoDragDropPicturesFromLinks, rvoAlt0CodesUseKeyboardCodepage,
    rvoFastDeleteProtectedText);
```

```
  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 [168] (special item type), depending on the value of SpacesInTab [761] property of the linked TRVStyle component.
rvoNoImageResize	Prevents resizing pictures [169] and controls [174] with mouse.
rvoAlt0CodesUseKeyboardCodepage	If not set (default), users can hold **Alt** and type a decimal Unicode character code on the numeric keypad, starting from '0' (**Num Lock** must be on), like in Microsoft Word If set, users can hold **Alt** and type a decimal ANSI character code on the numeric keypad, starting from '0', like most editors in Windows (the ANSI code page is determined by the active keyboard language).

Visual options

Option	Meaning
rvoNoCaretHighlightJumps	If set, hyperlinks will not be highlighted (with HoverColor [812] and HoverBackColor [811]) when the caret is moved inside them.
rvoHideReadOnlyCaret	If set, the editor hides the caret in read-only mode [488]

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 386 event. If *rvoAssignImageFileNames* in Options 247, 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 390 and OnRVMouseMove 403) when the user presses and holds **Ctrl** * key The editor switches back to the editing mode when the user releases **Ctrl** * key or changes document, or if the editor loses focus. * on macOS, **Command** is used instead of **Ctrl**.
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.

Protection options

Option	Meaning
rvoFastDeleteProtectedText	Specifies how a selected fragment that contains a modify-protected* text partially is deleted. If not set (default), deletion is not allowed (the user must select the protected text completely to delete it).

Option	Meaning
	If set, the selected fragment is deleted together with all the protected text (on deletion, the selection is automatically expanded to include the protected text completely).

* the protected text's style has *rvprModifyProtect* in Protection[815], and does not have *rvprDeleteProtect* option there.

Other options

Option	Meaning
rvoClearTagOnStyleApp	If set, editor clears *tags*[96] (set to '') of all text items when changing their style index using ApplyTextStyle[502] or ApplyStyleConversion[501] method.
rvoAutoSwitchLang	Obsolete, does nothing.

Default value (changed in v1.9):

[*rvoWantTabs*, *rvoCtrlJumps*]

See also:

- Options[247].

See also events:

- OnJump[390];
- OnRVMouseMove[403].

See also:

- TRVStyle.SpacesInTab[761];
- RichView Hypertext[97];
- *Tags*[96].

6.1.2.1.11 TRichViewEdit.ForceFieldHighlight

Allows highlighting label items[181] (including numbered sequences[183], footnotes[185], endnotes[184], sidenotes[186]) even when the editor is not focused.

```
property ForceFieldHighlight: Boolean;
```

(Introduced in version 15)

The editor can highlight label items with Style[260].FieldHighlightColor[746].

There are three modes possible, depending on Style[260].FieldHighlightType[747]: 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.

6.1.2.1.12 TRichViewEdit.KeyboardType

Determines the type of the virtual keyboard.

```
property KeyboardType: TVirtualKeyboardType;
```

(Introduced in version 23)

Default value

TVirtualKeyboardType.Default

6.1.2.1.13 TRichViewEdit.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 [374].
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 TRichView [138].

6.1.2.1.14 TRichViewEdit.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 [286] resets it to *False*.

Any editing operation, including calling any editor-style method (which invokes OnChange [582] 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.

6.1.2.1.15 TRichViewEdit.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[480] 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[372].

If the caret is inside table cell, this property returns 1 (position after the table). Items of subdocuments (table cells[951]) 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[489].CurItemNo[480] and TopLevelEditor[489].OffsetInCurItem.

This property must be accessed only when the document is formatted.

See also properties:

- CurItemNo[480];
- TopLevelEditor[489].

See also events:

- OnCaretMove[582].

See also methods:

- SetSelectionBounds[372];
- GetSelectionBounds[320].

See also:

- Inserting at the position of the caret[119];
- How to: move the caret to the beginning or to the end of document in TRichViewEdit[1133].

6.1.2.1.16 TRichViewEdit.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[483].

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 [582].

6.1.2.1.17 TRichViewEdit.SmartPopupProperties

Properties for "smart popups" (analog of Microsoft Office's "smart tags" buttons).

```
property SmartPopupProperties: TRVSmartPopupProperties [598];
```

(introduced in version 10)

See TRVSmartPopupProperties [598] for details.

See also properties:

- SmartPopupVisible [489].

See also events:

- OnSmartPopupClick [595].

6.1.2.1.18 TRichViewEdit.SmartPopupVisible

Shows/hides "smart popup" [598] 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 [582] 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 [489].

See also events:

- OnSmartPopupClick [595].

6.1.2.1.19 TRichViewEdit.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 [872] 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.

6.1.2.1.20 TRichViewEdit.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 [120].

6.1.2.2 Methods

Derived from TCustomRichViewEdit

AddSpellingMenuItems [497] [FMX]
AdjustControlPlacement [497]
AdjustControlPlacement2 [497]
ApplyListStyle [498]
ApplyParaStyle [498]
ApplyParaStyleConversion [499]
ApplyParaStyleTemplate [500]
ApplyStyleConversion [501]
ApplyStyleTemplate [501]
ApplyTextStyle [502]
ApplyTextStyleTemplate [502]
BeginItemModify [503]
BeginUndoCustomGroup [504]

BeginUndoCustomGroup2 504
BeginUndoGroup 505
BeginUndoGroup2 505
BeginUpdate 506
CanChange 506
CanPaste 507
CanPasteHTML 507
CanPasteRTF 507
CanPasteRVF 507
Change 508
ChangeListLevels 508
ChangeStyleTemplates 508
Clear 509
ClearTextFlow 509
ClearUndo 509
ConvertToHotPicture 510
ConvertToPicture 510
Create 510
CutDef 511
DeleteSelection 511
Destroy 511
EndItemModify 511
EndUndoGroup2 505
EndUndoCustomGroup2 504
EndUpdate 512
GetCheckpointAtCaret 512
GetCurrentBreakInfo 512
GetCurrentBulletInfo 513
GetCurrentCheckpoint 514
GetCurrentControlInfo 515
GetCurrentHotspotInfo 515
GetCurrentItemExtraIntProperty 518
GetCurrentItemExtraIntPropertyEx 518
GetCurrentItemExtraStrProperty 519
GetCurrentItemExtraStrPropertyEx 519
GetCurrentItem 517
GetCurrentItemEx 517
GetCurrentItemText -A -W 519
GetCurrentItemVAlign 520
GetCurrentLineCol 521
GetCurrentMisspelling 521
GetCurrentPictureInfo 522
GetCurrentTag 522
GetCurrentTextInfo 523
InsertBreak 523
InsertBullet 524
InsertCheckpoint 525

InsertControl 526
InsertDocXFromFileEd 526 (D2009+ or with Delphi ZLib)
InsertDocXFromStreamEd 527 (D2009+ or with Delphi ZLib)
InsertHotPicture 528
InsertHotspot 528
InsertHTMLFromFileEd 529
InsertHTMLFromStreamEd 530
InsertItem 531
InsertMarkdownFromFileEd 532
InsertMarkdownFromStreamEd 532
InsertOEMTextFromFile 533
InsertPageBreak 534
InsertPicture 534
InsertRTFFromFileEd 536
InsertRTFFromStreamEd 536
InsertRVFFromFileEd 537
InsertRVFFromStreamEd 538
InsertStringTag -A -W 539
InsertTab 540
InsertText -A -W 540
InsertTextFromFile -W 541
MoveCaret 542
Paste 543
PasteBitmap 544
PasteHTML 545
PasteGraphicFile 544
PasteMarkdown 546
PasteMetafile 546
PasteRTF 547
PasteRVF 547
PasteText -A -W 548
PasteURL 549
Redo 549
RedoAction 549
RedoName 550
RemoveCheckpointAtCaret 550
RemoveCheckpointEd 550
RemoveCurrentCheckpoint 551
RemoveCurrentPageBreak 551
RemoveLists 552
ResizeControl 552
ResizeCurrentControl 552
SearchText -A -W 553
SelectCurrentLine 554
SelectCurrentWord 555

SetBackgroundImageEd 556
SetBackgroundPropertyEd 557
SetBreakInfoEd 557
SetBulletInfoEd 558
SetCheckpointInfoEd 560
SetControlInfoEd 560
SetCurrentBreakInfo 561
SetCurrentBulletInfo 562
SetCurrentCheckpointInfo 563
SetCurrentControlInfo 564
SetCurrentHotspotInfo 565
SetCurrentItemExtraIntProperty 566
SetCurrentItemExtraIntPropertyEx 566
SetCurrentItemExtraStrProperty 567
SetCurrentItemExtraStrPropertyEx 567
SetCurrentItemText -A -W 568
SetCurrentItemVAlign 569
SetCurrentPictureInfo 569
SetCurrentTag 570
SetFloatPropertyEd 555
SetHotspotInfoEd 570
SetIntPropertyEd 555
SetItemExtraIntPropertyEd 572
SetItemExtraIntPropertyExEd 572
SetItemExtraStrPropertyEd 573
SetItemExtraStrPropertyExEd 573
SetItemTagEd 574
SetItemTextEd -A -W 574
SetItemVAlignEd 575
SetPictureInfoEd 576
SetStrPropertyEd 555
SetUndoGroupMode 577
Undo 577
UndoAction 577
UndoName 578

Derived from TCustomRichView 216

AddBreak 269
AddBullet 270
AddCheckpoint 271
AddControl 272
AddFmt 273
AddHotPicture 274
AddHotspot 275
AddItem 276

AddNL -A -W 277
AddPicture 278
AddTab 280
AddTextNL -A -W 281
AppendFrom 282
AppendRVFFromStream 283
AssignSoftPageBreaks 284
BeginOleDrag 284
Clear 286
ClearLiveSpellingResults 286
ClearSoftPageBreaks 286
ClientToDocument, DocumentToClient 286
ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits 287
Copy 288
CopyDef 288
CopyImage 289
CopyMarkdown 289
CopyRTF 290
CopyRVF 290
CopyText -A -W 290
Create 291
DeleteItems 291
DeleteMarkedStyles 292
DeleteParas 293
DeleteSection 293
DeleteUnusedStyles 293
Deselect 294
Destroy 294
FindCheckPointByName 294
FindCheckPointByTag 295
FindControlItemNo 295
Format 295
FormatTail 296
GetBreakInfo 296
GetBulletInfo 297
GetCheckpointByNo 298
GetCheckpointInfo 299
GetCheckpointItemNo 299
GetCheckpointNo 299
GetCheckpointXY 300
GetCheckpointY 300
GetCheckpointYEx 301
GetControlInfo 301
GetFirstCheckpoint 302
GetFocusedItem 302
GetHotspotInfo 303
GetItem 304

GetItemAt 305
GetItemCheckpoint 305
GetItemCoords 306
GetItemCoordsEx 307
GetItemExtraIntProperty 307
GetItemExtraIntPropertyEx 307
GetItemExtraStrProperty 308
GetItemExtraStrPropertyEx 308
GetItemNo 309
GetItemPara 309
GetItemStyle 310
GetItemTag 310
GetItemText -A -W 310
GetItemVAlign 311
GetJumpPointLocation 312
GetJumpPointY 313
GetLastCheckpoint 313
GetLineNo 314
GetListMarkerInfo 315
GetNextCheckpoint 315
GetOffsAfterItem 316
GetOffsBeforeItem 317
GetPictureInfo 318
GetPrevCheckpoint 319
GetRealDocumentPixelsPerInch 319
GetSelectedImage 319
GetSelectionBounds 320
GetSelText -W 321
GetTextInfo 321
GetWordAt -A -W 322
InsertRVFFromStream 323
IsFromNewLine 325
IsParaStart 325
LiveSpellingValidateWord 326
LoadDocX 326 (D2009+ or with Delphi ZLib)
LoadDocXFromStream 327 (D2009+ or with Delphi ZLib)
LoadFromFile 328
LoadFromFileEx 328
LoadFromStream 329
LoadFromStreamEx 329
LoadHTML 330
LoadHTMLFromStream 332
LoadMarkdown 332
LoadMarkdownFromStream 333
LoadRTF 334
LoadRTFFromStream 335
LoadRVF 336

LoadRVFFromStream 337
LoadText -W 338
LoadTextFromStream -W 339
MarkStylesInUse 339
RefreshListMarkers 340
Reformat 339
RemoveCheckpoint 340
ResetAnimation 341
SaveDocX 341
SaveDocXToStream 342
SaveHTML 342
SaveHTMLToStream 346
SavePicture 349
SaveRTF 350
SaveRTFToStream 351
SaveRVF 351
SaveRVFToStream 352
SaveText -W 352
SaveTextToStream -W 353
SearchText -A -W 354
SelectAll 355
SelectControl 355
SelectionExists 356
SelectWordAt 356
SetAddParagraphMode 357
SetBreakInfo 358
SetBulletInfo 359
SetCheckpointInfo 360
SetControlInfo 361
SetFooter 362
SetHeader 363
SetHotspotInfo 364
SetItemExtraIntProperty 366
SetItemExtraIntPropertyEx 366
SetItemExtraStrProperty 367
SetItemExtraStrPropertyEx 367
SetItemTag 367
SetItemText -A -W 368
SetItemVAlign 369
SetListMarkerInfo 369
SetPictureInfo 370
SetSelectionBounds 372
StartAnimation 374
StartLiveSpelling 374
StopAnimation 375
StoreSearchResult 375
UpdatePaletteInfo 375

Derived from TRVScroller [869]

ScrollTo [875]

6.1.2.2.1 TRichViewEdit.AddSpellingMenuItems

Adds menu items offering corrections for the current misspelled word to **AMenu**.

```
procedure AddSpellingMenuItems(AMenu: TCustomPopupMenu);
```

If PopupMenu property is not assigned, the editor uses a built-in standard popup menu. Spelling checking items are added in this standard menu automatically, and this method is not needed.

If you use your own popup menu, you can call this method to add items that offer corrections for the current misspelled word. Before adding new items, the method destroys all items that were added by the previous call of this method.

If the current word is not misspelled, no items are added (but previously added items are destroyed).

Suggestions are requested:

- from OnGetSpellingSuggestions [382] event;
- from a platform spelling check service, if it is available for the current platform.

6.1.2.2.2 TRichViewEdit.AdjustControlPlacement

Returns the control [174] 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 [552] (or ResizeControl [552]) 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 [497] instead of this method, because it can work with controls inserted in table [196] cells.

This method must be called only when the document is formatted.

6.1.2.2.3 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 [552] (or ResizeControl [552]) 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 [497].

6.1.2.2.4 TRichViewEdit.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 [260].ListStyles [755])

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[**AListNo**].Levels [842]).

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 [369]).

See also:

- RemoveLists [552];
- ChangeListLevels [508].

See also methods of TCustomRichView:

- SetListMarkerInfo [369];
- GetListMarkerInfo [315].

6.1.2.2.5 TRichViewEdit.ApplyParaStyle

Applies the specified paragraph style.

```
procedure ApplyParaStyle(ParaStyleNo: Integer);
```

ParaStyleNo – index in the collection of paragraph styles (Style [260].ParaStyles [757]).

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 [833], this method does not change their styles.

Method type: editing-style.

If you want to implement commands like "change alignment", "increase indent" use ApplyParaStyleConversion [499] instead.

See also properties:

- CurParaStyleNo [481].

See also methods:

- ApplyTextStyle [502];
- ApplyParaStyleConversion [499].

See also:

- TParaInfo.Options [833];
- Paragraphs overview [100].

6.1.2.2.6 TRichViewEdit.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 [592] event for each selected item. Using this method, you can implement commands like "change alignment" or "increase indent".

This method ignores any paragraph protection [833], it can change paragraph style even if *rvpaoStyleProtect* is included in Options [833] of existing paragraphs' styles. If you want to respect paragraph style protection, check this option yourself in OnParaStyleConversion [592] event.

If **Recursive** = *False*, OnParaStyleConversion [592] is not called for cells of selected tables [179] (except for the case of multicell selection in a single table).

Method type: editing-style.

Demo:

- Demos*\Editors\Editor 2\

See also:

- ApplyStyleConversion [501];
- ApplyParaStyle [498].

See also events:

- OnParaStyleConversion [592].

See also:

- Paragraphs overview [100].

6.1.2.2.7 TRichViewEdit.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 [757] of items in the selected paragraphs.

Parameters:

TemplateNo – index in the collection Style [260].StyleTemplates [762], or the value -1. The value -1 means a format clearing (see below). The style template must have Kind [845] = *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 [179] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates [263]=*False* and **TemplateNo**>=0.

If paragraphs are formatted with styles having *rvpaoStyleProtect* in their Options [833], 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 [805]=-1.

If UseStyleTemplates [263]=*False* and **TemplateNo**<0, the results are identical to calling ApplyParaStyle [498](0).

Modified properties

Properties of a paragraph style [837] are considered modified, if they have values different from values of the corresponding properties of the current style template (referred in StyleTemplateId [805] property). If a style template is not assigned, the properties are considered modified if they have non-default values. The method ignores ModifiedProperties [839] property (it calculates this property itself).

Method type: editing-style.

See also:

- ApplyTextStyleTemplate [502];
- ApplyParaStyleConversion [499];
- ApplyParaStyle [498].

6.1.2.2.8 TRichViewEdit.ApplyStyleConversion

Applies a custom conversion procedure to styles of the selected text items [167], tabs [168], label items [181] (and item of all types inherited from label items).

```
procedure ApplyStyleConversion(UserData: Integer;
  Recursive: Boolean = True);
```

This procedure calls OnStyleConversion [595] event for each selected text item and for the current text style [482]. Using this method, you can implement commands like "make bold" or "change font name".

This method ignores any text protection [815], it can change text style even if *rvprStyleProtect* is included in Protection [815] of existing text items' styles. If you want to respect text style protection, check this protection option yourself in OnStyleConversion [595] event.

If **Recursive** = *False*, OnStyleConversion [595] is not called for cells of selected tables [179] (except for the case of multicell selection in a single table).

Method type: editing-style.

Demo:

- Demos*\Editors\Editor 2\

See also:

- ApplyParaStyleConversion [499];
- ApplyTextStyle [502].

See also events:

- OnStyleConversion [595].

6.1.2.2.9 TRichViewEdit.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 [764] or paragraph styles [757] of items in the selected paragraphs. The result depends on the style template's Kind [845] and the selection.

Parameters:

TemplateNo – index in the collection Style [260].StyleTemplates [762], 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 [179] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates [263]=*False* and **TemplateNo**>=0.

The method works according to the following rules:

- if Kind [845] of the specified style template = *rvstkText*, it is applied to the selected text;
- if Kind [845] of the specified style template = *rvstkPara*, it is applied to the selected paragraphs; the only exception is "Normal" style template [799];
- if **TemplateNo**=-1, or Kind [845] 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[502] or ApplyParaStyleTemplate[500] called with **ResetAdditionalFormatting**=*True*.

The method skips paragraph styles having *rvpaoStyleProtect* in their Options[833], and text styles having *rvprStyleProtect* in their Protection[815].

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[502] and ApplyParaStyleTemplate[500].

Method type: editing-style.

See also:

- ChangeStyleTemplates[508].

6.1.2.2.10 TRichViewEdit.ApplyTextStyle

Applies the specified text style to the selected items (text items[167], tabs[168], label items[181])

```
procedure ApplyTextStyle(TextStyleNo: Integer);
```

TextStyleNo – index in the collection of text styles (Style[260].TextStyles[764]).

If selection is empty, this method only sets CurTextStyleNo[482] to **TextStyleNo**. If the selection is not empty, the method changes style of text items[167] in the selection.

If the existing text items are formatted with styles having *rvprStyleProtect* in their Protection[815], 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[501] instead of this method.

See also properties:

- CurTextStyleNo[482];
- EditorOptions[483].

See also methods:

- ApplyParaStyle[498].

6.1.2.2.11 TRichViewEdit.ApplyTextStyleTemplate

Applies the specified style template to the selected text items[167], tabs[168], label items[181] (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[764] of the selected items.

Parameters:

TemplateNo – index in the collection Style[260].StyleTemplates[762], or the value -1. The value -1 means a format clearing (see below). The style template must have Kind[845] = *rvstkParaText* or *rvstkText*. The only exception is "Normal" style template[799]: 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[179] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates[263]=*False* and **TemplateNo**>=0.

If items are formatted with text styles having *rvprStyleProtect* in their Protection[815], this method skips them.

Format clearing

A format clearing means resetting all text attributes to default values. If a paragraph style template (ParaStyleTemplateId[815]) is assigned, it defines these default values. A special processing is performed for hyperlinks (i.e. text styles with Jump[823]=*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[263]=*False* and **TemplateNo**<0, the results are identical to calling ApplyTextStyle[502](0).

Modified properties

Properties of a text style[821] are considered modified, if they have values different from values of the corresponding properties of the current style templates (referred in StyleTemplateId[805] and ParaStyleTemplateId[815] properties). If none of these style templates are assigned, the properties are considered modified of they have non-default values. The method ignores ModifiedProperties[824] property (it calculates this property itself).

Method type: editing-style.

See also:

- ApplyParaStyleTemplate[500];
- ApplyStyleConversion[501];
- ApplyTextStyle[502].

6.1.2.2.12 TRichViewEdit.BeginItemModify

Stores some information required for quick reformatting after modification of item.

```
procedure BeginItemModify(ItemNo: Integer; out ModifyData: TRVCoord[1064]);
```

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[511] after performing modifications.

Usually this function is used when modifying tables[901].

This method must be called only when the document is formatted.

Example is in Table Operations Overview[205]

6.1.2.2.13 TRichViewEdit.BeginUndoCustomGroup

Begins a new undo action.

```
procedure BeginUndoCustomGroup(const Name: TRVUnicodeString[1099]);
```

(changed in version 18)

Begins a new undo action.

Usually followed by SetUndoGroupMode[577](*True*) ... SetUndoGroupMode(*False*), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as BeginUndoGroup[505], 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[489]).

See also:

- Undo in RichViewEdit[120].

6.1.2.2.14 TRichViewEdit.BeginUndoCustomGroup2, EndUndoCustomGroup2

An alternative method to group operation for undo.

```
procedure BeginUndoCustomGroup2(const Name: TRVUnicodeString[1099]);
procedure EndUndoCustomGroup2(const Name: TRVUnicodeString[1099]);
```

(introduced in version 18)

The recommended and most efficient way to group operations (to undo them in a single step) is:

```
Edit.TopLevelEditor[489].BeginUndoCustomGroup[504](OperationName);
Edit.TopLevelEditor.SetUndoGroupMode[577](True);
<operations>
Edit.TopLevelEditor.SetUndoGroupMode(False);
```

BeginUndoCustomGroup2 and **EndUndoCustomGroup2** provide an alternative way to group operations. Call **BeginUndoCustomGroup2**, then operations you want to group, then **EndUndoCustomGroup2** (with the same Name):

```
Edit.BeginUndoGroup2(OperationName);
<operations>
Edit.EndUndoGroup2(OperationName);
```

BeginUndoCustomGroup2 and **EndUndoCustomGroup2** do the same work as BeginUndoGroup2 and EndUndoGroup2[505], but allows to give a custom name to this group of changes.

This way of grouping is less efficient and requires more memory, but it allows grouping operations performed in several table cells (for example, replacing all occurrences of text).

These methods require an unlimited undo buffer (UndoLimit[490] must be equal to -1), otherwise they do nothing.

Unlike the first set of method (which must be called for TopLevelEditor[489]), these method must be called for a root editor.

Calls of **BeginUndoCustomGroup2..EndUndoCustomGroup2** may be nested, but it is not recommended. You can use BeginUndoGroup, BeginUndoCustomGroup, and SetUndoGroupMode inside **BeginUndoCustomGroup2..EndUndoCustomGroup2**, but not vice versa.

See also:

- Undo in RichViewEdit[120]

6.1.2.2.15 TRichViewEdit.BeginUndoGroup

Begins a new undo action.

```
procedure BeginUndoGroup(UndoType: TRVUndoType[1098]);
```

Begins a new undo action.

Usually followed by SetUndoGroupMode[577](*True*) ... SetUndoGroupMode(*False*), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as BeginUndoCustomGroup[504], 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[489]).

See also:

- Undo in RichViewEdit[120], includes an example.

6.1.2.2.16 TRichViewEdit.BeginUndoGroup2, EndUndoGroup2

An alternative method to group operations for undo.

```
procedure BeginUndoGroup2(UndoType: TRVUndoType[1098]);
procedure EndUndoGroup2(UndoType: TRVUndoType[1098]);
```

(introduced in version 18)

The recommended and most efficient way to group operations (to undo them in a single step) is:

```
Edit.TopLevelEditor[489].BeginUndoGroup[505](UndoType);
Edit.TopLevelEditor.SetUndoGroupMode[577](True);
<operations>
Edit.TopLevelEditor.SetUndoGroupMode(False);
```

BeginUndoGroup2 and **EndUndoGroup2** provide an alternative way to group operations. Call **BeginUndoGroup2**, then operations you want to group, then **EndUndoGroup2** (with the same UndoType):

```
Edit.BeginUndoGroup2(UndoType);
<operations>
Edit.EndUndoGroup2(UndoType);
```

BeginUndoGroup2 and **EndUndoGroup2** do the same work as BeginUndoCustomGroup2 and EndUndoCustomGroup2 [504], but give a standard name (undo identifier) to this group of changes.

This way of grouping is less efficient and requires more memory, but it allows grouping operations performed in several table cells (for example, replacing all occurrences of text).

These methods require an unlimited undo buffer (UndoLimit [490] must be equal to -1), otherwise they do nothing.

Unlike the first set of methods (which must be called for TopLevelEditor [489]), these method must be called for a root editor.

Calls of **BeginUndoGroup2..EndUndoGroup2** may be nested, but it is not recommended. You can use BeginUndoGroup, BeginUndoCustomGroup, and SetUndoGroupMode inside **BeginUndoGroup2..EndUndoGroup2**, but not vice versa.

See also:

- Undo in RichViewEdit [120], includes an example.

6.1.2.2.17 TRichViewEdit.BeginUpdate

Suspends repainting until EndUpdate [512] 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.

6.1.2.2.18 TRichViewEdit.CanChange

Returns: "can the document in this editor be changed now?"

```
function CanChange: Boolean;
```

This function checks ReadOnly [488] property.

Usually this function is called before performing operations on tables [901] to prevent modifying editor in read-only state.

Additionally, if live spelling checking [138] is active, this method pauses a live spelling checking thread (until the call of Change [508]), to prevent errors when accessing data that are being modified.

For TDBRichViewEdit [616], 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.

6.1.2.2.19 TRichViewEdit.CanPaste

Returns *True* if

- the Clipboard contains data in at least one format which RichViewEdit can paste
- this format is included in AcceptPasteFormats [478] property.

```
function CanPaste: Boolean;
```

See also methods:

- Paste [543];
- CanPasteRVF [507];
- CanPasteHTML [507];
- CanPasteRTF [507].

See also:

- Working with Clipboard [113].

6.1.2.2.20 TRichViewEdit.CanPasteHTML

Returns *True* if the Clipboard contains data in HTML format.

```
function CanPasteHTML: Boolean;
```

(Introduced in version 21)

Platform notes: this method is implemented for Windows and macOS. For other platforms, it returns *False*.

See also methods:

- PasteHTML [545];
- CanPaste [507].

See also:

- Working with Clipboard [113]

6.1.2.2.21 TRichViewEdit.CanPasteRTF

Returns *True* if the Clipboard contains data in RTF (Rich Text Format).

```
function CanPasteRTF: Boolean;
```

(Introduced in version 1.5)

See also methods:

- PasteRTF [547];
- CanPaste [507].

See also:

- Working with Clipboard [113]

6.1.2.2.22 TRichViewEdit.CanPasteRVF

Returns *True* if the Clipboard contains data in RVF (RichView Format [129]).

```
function CanPasteRVF: Boolean;
```

See also methods:

- PasteRVF [547];
- CanPaste [507].

See also:

- Working with Clipboard[113];
- RVF[129].

6.1.2.2.23 TRichViewEdit.Change

Generates OnChange[582] event.

```
procedure Change;
```

This method also assigns *True* to Modified[487] property.

6.1.2.2.24 TRichViewEdit.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[498];
- RemoveLists[552].

See also methods of TCustomRichView:

- SetListMarkerInfo[369];
- GetListMarkerInfo[315].

6.1.2.2.25 TRichViewEdit.ChangeStyleTemplates

Updates the document after editing style templates.

```
procedure ChangeStyleTemplates(NewStyleTemplates:
  TRVStyleTemplateCollection[797]);
```

(introduced in version 14)

The code below shows how to edit style templates for RichViewEdit1: TRichViewEdit. It assumes that you have RVStyleTmp: TRVStyle[739] component, not linked to any RichView.

```
RVStyleTmp.Units[764] := RichViewEdit1.Style.Units;
RVStyleTmp.StyleTemplates[762].AssignStyleTemplates[799](
  RichViewEdit1.Style.StyleTemplates, True);
<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[798]).

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[501];
- ApplyParaStyleTemplate[500];
- ApplyTextStyleTemplate[502].

6.1.2.2.26 TRichViewEdit.Clear

Clears the document.

```
procedure Clear;override;
```

This method:

- deletes all content of the document (see RichView.Clear[286]);
- sets the current text style index (CurTextStyleNo[482]) and the current paragraph style index (CurParaStyleNo[481]) to zeros;
- clears undo and redo buffers (see ClearUndo[509]);
- sets Modified[487] to *False*.

This method does not check ReadOnly[488], does not reformat and repaint component, does not generate OnChange[582].

This methods makes the document unformatted.

6.1.2.2.27 TRichViewEdit.ClearTextFlow

Clears text flow around left- and right-aligned items.

```
procedure ClearTextFlow(Left, Right: Boolean);
```

(introduced in version 12)

This method changes ClearLeft[232] and ClearRight[232] property for the first selected paragraph.

Method type: editing-style.

See also:

- TRVVAlign[1100] type (*rvvaLeft* and *rvvaRight*)

6.1.2.2.28 TRichViewEdit.ClearUndo

Clears undo and redo buffers of the editor.

```
procedure ClearUndo;
```

(Introduced in version 1.3)

See also:

- Undo in RichViewEdit[120].

6.1.2.2.29 TRichViewEdit.ConvertToHotPicture

Converts the item type from picture [169] to *hot-picture* [172].

```
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 [244]-1) it works with the specified item

If this item is not a picture (GetItemStyle [310]/CurItemStyle [481] <> *rvsPicture*), this method does nothing.

Method type: editing-style.

See also:

- ConvertToPicture [510].

6.1.2.2.30 TRichViewEdit.ConvertToPicture

Converts the item type from *hot-picture* [172] to picture [169]

```
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 [244]-1) it works with the specified item.

If this item is not a *hot-picture* (GetItemStyle [310]/CurItemStyle [481] <> *rvsHotPicture*), this method does nothing.

Method type: editing-style.

See also:

- ConvertToHotPicture [510].

6.1.2.2.31 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 [224] (and overridden by default!)

6.1.2.2.32 TRichViewEdit.CutDef

Cuts selection to the Clipboard

```
procedure CutDef;
```

This method copies selection to the Clipboard using CopyDef[288] method. If the selection was copied (at least one of copy formats is selected in Options[247]), it calls DeleteSelection[511].

This method is executed automatically when user presses Ctrl+X (Command+X on macOS) or Shift+Delete.

Method type: editing-style.

See also methods:

- CopyDef[288];
- Paste[543];
- SelectionExists[356].

See also:

- Working with selection[112];
- Working with Clipboard[113].

6.1.2.2.33 TRichViewEdit.DeleteSelection

Deletes the selected fragment in the editor.

```
procedure DeleteSelection;
```

Method type: editing-style.

See also:

- Working with selection[112].

6.1.2.2.34 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.

6.1.2.2.35 TRichViewEdit.EndItemModify

Quickly reformats editor after modifying item.

```
procedure EndItemModify(ItemNo: Integer; ModifyData: TRVCoord[1064]);
```

Parameters:

ItemNo – index of item which was modified.

ModifyData – the value obtained in BeginItemModify[503]

Usually this function is used when modifying tables[901].

Example is in Table Operations Overview[205]

6.1.2.2.36 TRichViewEdit.EndUpdate

Reenables editor repainting that was turned off with the BeginUpdate[506] 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.

6.1.2.2.37 TRichViewEdit.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[514].

Caret Position	GetCheckpointAtCaret	GetCurrentCheckpoint[514]
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[525] and RemoveCheckpointAtCaret[550]. The alternative set of methods is: SetCurrentCheckpointInfo[563], GetCurrentCheckpoint[514] and RemoveCurrentCheckpoint[551].

See also:

- "Checkpoints"[92].

6.1.2.2.38 TRichViewEdit.GetCurrentBreakInfo

Returns main properties for the item of *break*[173] type at the position of caret.

```
procedure GetCurrentBreakInfo(out AWidth: TRVStyleLength[1094];
  out AStyle: TRVBreakStyle[1059]; out AColor: TRVColor[1063];
  out ATag: TRVTag[1096]);
```

GetCurrentBreakInfo(...) is equivalent to TopLevelEditor[489].**GetBreakInfo**[296](TopLevelEditor[489].CurItemNo[480], ...).

The item must be of *break*[173] type (*rvsBreak*[1127]), otherwise the method raises ERichViewError[1014] exception. Type of item at the position of caret is returned by CurItemStyle[481].

Output parameters:

AWidth – line width (or rectangle height). This value is measured in Style[260].Units[764].

AStyle – visual style of this *break*, see TRVBreakStyle[1059] for possible values.

AColor – line color. If it is equal to *rvclNone*[1105], Style[260].TextStyles[764][0].Color[810] is used.

ATag – *tag* of the item. Use SetCurrentTag[570] or SetCurrentBreakInfo[561] to change *tag* of item as an editing operation.

Instead of this method, you can use GetCurrentItemExtraIntPropertyEx[518] and GetCurrentTag[522] 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[518] and GetCurrentItemExtraStrProperty[519].

See also methods:

- SetCurrentBreakInfo[561] (changes properties of *break* at the position of caret, as an editing operation);
- GetBreakInfo[296] (returns properties of the specified *break*).

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.2.2.39 TRichViewEdit.GetCurrentBulletInfo

Returns main properties for the item of *bullet*[176] type at the position of caret.

VCL and LCL:

```
procedure GetCurrentBulletInfo(out AName: TRVUnicodeString[1099];
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag[1096]);
```

FireMonkey:

```
procedure GetCurrentBulletInfo(out AName: TRVUnicodeString[1099];
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag[1096];
  out AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

GetCurrentBulletInfo(...) is equivalent to TopLevelEditor[489].**GetBulletInfo**[297](TopLevelEditor[489].CurItemNo[480], ...).

Output parameters:

AName – name of *bullet*[176]. It can also be read using GetCurrentItemText[519] method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using GetCurrentItemExtraIntPropertyEx[518] method.

ATag – *tag* of the item. Use SetCurrentTag[570] or SetCurrentBulletInfo[562] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[522] method.

Additional output parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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[518] and GetCurrentItemExtraStrProperty[519].

See also methods:

- SetCurrentBulletInfo[562] (changes properties of *bullet* at the position of caret, as an editing operation);
- GetBulletInfo[297] (returns properties of the specified *bullet*).

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.2.2.40 TRichViewEdit.GetCurrentCheckpoint

Returns *checkpoint* associated with the item at position of caret, or *nil* if this item does not have associated *checkpoint*.

```
function GetCurrentCheckpoint: TCheckPointData[1055];
```

The same as TopLevelEditor[489].**GetItemCheckpoint**[305](TopLevelEditor[489].CurItemNo[480]).

This method must be called only when the document is formatted.

See also methods:

- SetCurrentCheckpointInfo[563] (changes *checkpoint* of item at the position of caret, as an editing operation);
- GetItemCheckpoint[305] (returns *checkpoint* of the specified item).

See also properties:

- CurItemNo[480].

See also:

- "Checkpoints"[92].

6.1.2.2.41 TRichViewEdit.GetCurrentControlInfo

Returns main properties for the item of inserted control [174] type at position of caret.

```
procedure GetCurrentControlInfo(out AName: TRVUnicodeString[1099];
  out Actrl: TControl; out AVAlign: TRVVAlign[1100];
  out ATag: TRVTag[1096]);
```

(changed in version 18)

GetCurrentControlInfo(...) is equivalent to TopLevelEditor[489].**GetControlInfo**[301](TopLevelEditor[489].CurItemNo[480], ...).

Output parameters:

AName – name of control item[174]. Do not confuse with **Actrl**.Name property! This value can also be read using GetCurrentItemText[519] 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[570] or SetCurrentControlInfo[564] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[522] 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[518] and GetCurrentItemExtraStrProperty[519].

See also methods:

- SetCurrentControlInfo[564] (changes properties of control item at the position of caret, as an editing operation);
- GetControlInfo[301] (returns properties of the specified control item).

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.2.2.42 TRichViewEdit.GetCurrentHotspotInfo

Returns main properties for the item of *hotspot*[178] type at the position of caret.

VCL and LCL:

```
procedure GetCurrentHotspotInfo(out AName: TRVUnicodeString[1099];
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure GetCurrentHotspotInfo(out AName: TRVUnicodeString[1099];
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag[1096];
  out AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

GetCurrentHotspotInfo(...) is equivalent to TopLevelEditor[489].**GetHotspotInfo**[303] (TopLevelEditor[489].CurItemNo[480], ...).

Output parameters:

AName – name of *hotspot*[178]. It can also be read using GetCurrentItemText[519] 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[518] 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[518] method.

ATag – *tag* of the item. Use SetCurrentTag[570] or SetCurrentHotspotInfo[565] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[522] method.

Additional output parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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[518] and GetCurrentItemExtraStrProperty[519].

See also methods:

- SetCurrentHotspotInfo[565] (changes properties of *hotspot* at the position of caret, as an editing operation);
- GetHotspotInfo[303] (returns properties of the specified *hotspot*).

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.2.2.43 TRichViewEdit.GetCurrentItem

Returns object representing item at position of caret.

```
function GetCurrentItem: TCustomRVItemInfo[899];
```

(introduced in version 1.4)

This method is usually used when working with tables[179], labels[181], sequences[183], footnotes[185], endnotes[184], references to notes[190].

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[517] instead of GetCurrentItem.

This method must be called only when the document is formatted.

When called for an empty editor (i.e. ItemCount[244]=0), the method returns *nil*.

See also methods:

- GetCurrentItemEx[517];
- InsertItem[531];
- GetItem[304].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Tables in RichView[196].

6.1.2.2.44 TRichViewEdit.GetCurrentItemEx

Returns object at the position of caret representing item of the given class.

```
function GetCurrentItemEx(RequiredClass: TCustomRVItemInfoClass[899];
  out ItemRichViewEdit: TCustomRichViewEdit;
  out Item: TCustomRVItemInfo[899]): Boolean;
```

(introduced in version 1.4)

This method is usually used when working with tables[196].

Input parameter:

RequiredClass – the required class of item, set it to TRVTableItemInfo[901].

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[517];
- InsertItem[531];
- GetItem[304].

See also:

- Obtaining RichView items[116];
- Tables in RichView[196].

6.1.2.2.45 TRichViewEdit.GetCurrentItemExtraIntProperty -Ex

The methods return a value of the specified integer property of the item at the position of caret

```
function GetCurrentItemExtraIntProperty(
  Prop: TRVExtraItemProperty[1067]; out Value: Integer): Boolean;
function GetCurrentItemExtraIntPropertyEx(
  Prop: Integer; out Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

GetCurrentItemExtraIntProperty[Ex](...) is equivalent to TopLevelEditor[489].**GetItemExtraIntProperty[Ex]**[307](TopLevelEditor[489].CurItemNo[480], ...).

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[1067]. 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[1119]

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][566] (set a value of an integer property for the item at the position of caret, as an editing operation);
- SetItemExtraIntProperty[Ex]Ed[572] (set a value of an integer property for the specified item, as an editing operation);

- GetCurrentItemExtra**Str**Property[519] (return a value of the specified *string* property of the item at the position of caret).

See also methods of TRichView

- GetItemExtraIntProperty[Ex][307].

6.1.2.2.46 TRichViewEdit.GetCurrentItemExtraStrProperty -Ex

The methods return a value of the specified string property of the item at the position of caret

```
function GetCurrentItemExtraStrProperty(
  Prop: TRVExtraItemStrProperty[1072];
  out Value: TRVUnicodeString[1099]): Boolean;
function GetCurrentItemExtraStrProperty(
  Prop: Integer; out Value: TRVUnicodeString[1099]): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

GetCurrentItemExtraStrProperty[Ex](...) is equivalent to TopLevelEditor[489].**GetItemExtraStrProperty[Ex]**[308](TopLevelEditor[489].CurItemNo[480], ...).

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[1072]. 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[1124]

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][567] (set a value of a string property for the item at the position of caret, as an editing operation);
- SetItemExtraStrProperty[Ex]Ed[573] (set a value of a string property for the specified item, as an editing operation);
- GetCurrentItemExtra**Int**Property[Ex][518] (return a value of the specified *integer* property of the item at the position of caret).

See also methods of TRichView:

- GetItemExtraStrProperty[Ex][308].

6.1.2.2.47 TRichViewEdit.GetCurrentItemText -A -W

Returns text for the item at the position of caret.

```
function GetCurrentItemText: String;
function GetCurrentItemTextA: TRVAnsiString[1056];
function GetCurrentItemTextW: TRVUnicodeString[1099];
```

(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[489].**GetItemText**[310](TopLevelEditor[489].CurItemNo[480]).
- **GetCurrentItemTextA** is the same as TopLevelEditor[489].**GetItemTextA**[310](TopLevelEditor[489].CurItemNo[480]).
- **GetCurrentItemTextW** is the same as TopLevelEditor[489].**GetItemTextW**[310](TopLevelEditor[489].CurItemNo[480]).

Unicode notes:

Internally, text is stored as Unicode. **GetCurrentItemTextA** converts Unicode to ANSI. For a text item, a code page for conversion is calculated basing on the its Charset[809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[260].DefCodePage[745] is used.

GetCurrentItemText returns:

- ANSI string, like **GetCurrentItemTextA**, in Delphi 2007 and older
- Unicode (UTF-16) string, like **GetCurrentItemTextW**, in Delphi 2009 and newer
- Unicode (UTF-8) string, in Lazarus[160]

These methods must be called only when the document is formatted.

See also methods:

- SetCurrentItemText -A -W[568].

See also methods of TCustomRichView:

- GetItemText -A -W[310].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96];
- Unicode in RichView[135].

6.1.2.2.48 TRichViewEdit.GetCurrentItemVAlign

Returns vertical alignment (position relative to the line) of the item at the caret position

```
function GetCurrentItemVAlign: TRVVAlign[1100];
```

(introduced in version 12)

The same as TopLevelEditor[489].**GetItemVAlign**[311](TopLevelEditor[489].CurItemNo[480]).

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[167], tables[179], *breaks*[173], tabs[168], list markers[180], footnotes[185] and endnotes[184].

See also methods:

- SetCurrentItemVAlign[569].

See also properties:

- CurItemNo[480].

6.1.2.2.49 TRichViewEdit.GetCurrentLineCol

Returns a line and column at the caret position.

```
procedure GetCurrentLineCol(out 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[314].

6.1.2.2.50 TRichViewEdit.GetCurrentMisspelling

Returns misspelled word at the position of caret, optionally selects it.

```
function GetCurrentMisspelling(SelectIt: Boolean;
  out Word: TRVUnicodeString[1099]; out 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[764] collection of the linked[260] 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[138].

6.1.2.2.51 TRichViewEdit.GetCurrentPictureInfo

Returns main properties for the item of picture[169]/*hot-picture*[172] type at the position of caret.

```
procedure GetCurrentPictureInfo(out AName: TRVUnicodeString[1099];
  out Agr: TRVGraphic[1032]; out AVAlign: TRVVAlign[1100]; out ATag: TRVTag[1096]);
```

(changed in version 18)

GetCurrentPictureInfo(...) is equivalent to TopLevelEditor[489].**GetPictureInfo**[318](TopLevelEditor[489].CurItemNo[480], ...).

Output parameters:

AName – name of the item. It can also be read using GetCurrentItemText[519] 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[570] or SetCurrentPictureInfo[569] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[522] 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[518] and GetCurrentItemExtraStrProperty[519].

See also methods:

- SetCurrentPictureInfo[569] (changes properties of picture at the position of caret, as an editing operation);
- GetPictureInfo[318] (returns properties of the specified picture).

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items[116];
- Item types[90];
- "Tags"[96].

6.1.2.2.52 TRichViewEdit.GetCurrentTag

Returns tag of the item at the caret position

```
function GetCurrentTag: TRVTag[1096];
```

The same as TopLevelEditor[489].**GetItemTag**[310](TopLevelEditor[489].CurItemNo[480]).

Use SetCurrentTag[570] 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 [570].

See also properties:

- CurItemNo [480].

See also

- "Tags" [96].

6.1.2.2.53 TRichViewEdit.GetCurrentTextInfo

Returns text and *tag* for text item at the position of caret

```
procedure GetCurrentTextInfo(out AText: TRVUnicodeString[1099];
  out ATag: TRVTag[1096]);
```

GetCurrentTextInfo(...) is equivalent to TopLevelEditor [489].**GetTextInfo** [321](TopLevelEditor [489].CurItemNo [480], ...).

Use SetCurrentTag [570] 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 [480];
- CurItemStyle [481].

See also:

- Obtaining items [116];
- Item types [90];
- "Tags" [96].

6.1.2.2.54 TRichViewEdit.InsertBreak

Inserts a *break* [173] (horizontal line) at the position of the caret.

```
function InsertBreak(AWidth: TRVStyleLength[1094]; AStyle: TRVBreakStyle[1059];
  AColor: TRVColor[1063]): Boolean;
```

Parameters:

AWidth – width of the line, measured in Style [260].Units [764];

AStyle – break type;

AColor – color of line. If it is equal to *rvclNone* [1105], the *break* has the color of the 0th text style [764].

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[269].

See also:

- Inserting items at position of caret[119];
- Item types[90].

6.1.2.2.55 TRichViewEdit.InsertBullet

Inserts a *bullet*[176] at the position of the caret.

VCL and LCL:

```
function InsertBullet(ImageIndex: Integer;
  ImageList: TCustomImageList): Boolean;
```

FireMonkey:

```
function InsertBullet(ImageIndex: Integer;
  ImageList: TCustomImageList;
  ImageWidth, ImageHeight: TRVStyleLength[1094]): 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.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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:

- AddBullet[270];

See also:

- Inserting items at position of caret[119];
- Item types[90].

6.1.2.2.56 TRichViewEdit.InsertCheckpoint

Inserts a *checkpoint* at the position of the caret.

```
function InsertCheckpoint(const ATag: TRVTag[1096];
  const AName: TRVUnicodeString[1099];
  ARaiseEvent: Boolean): Boolean;
```

(introduced in version 10; changed in versions 18 and 19)

This method is similar to SetCurrentCheckpointInfo[563], but inserts *checkpoint* exactly at the position of caret.

If the caret is at the beginning of item, this methods calls SetCheckpointInfoEd[560] to add *checkpoint* for this item.

If the caret is at the middle of text item[167], this method splits this item at the position of caret, and calls SetCheckpointInfoEd[560] 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[560] to add *checkpoint* for the first paragraph item (or the second item, if the first one is list marker[180]).

If the caret is at the end of document, this methods does nothing.

This method is used in conjunction with GetCheckpointAtCaret[512] and RemoveCheckpointAtCaret[550] (GetCurrentCheckpoint[514] and RemoveCurrentCheckpoint[551] 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[379] event for this *checkpoint*.

Return value:

True if the checkpoint was inserted. *False* otherwise (the function may fail because of a protection, a read-only mode, or when the caret is at the end of the document).

Method type: editing-style.

See also:

- Modifying RichView items[117];
- "Checkpoints"[92];
- "Tags"[96].

6.1.2.2.57 TRichViewEdit.InsertControl

Inserts a Delphi/C++Builder control [174] at the position of the caret.

```
function InsertControl(const Name: TRVUnicodeString[1099]; ctrl: TControl;
  VAlign: TRVVAlign[1100]): 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[1100] 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 [272];

See also:

- Inserting items at position of caret [119];
- Item types [90].

6.1.2.2.58 TRichViewEdit.InsertDocXFromFileEd

Inserts a DocX (Microsoft Word Document) from file **FileName** at the position of the caret.

```
function InsertDocXFromFileEd(
  const FileName: TRVUnicodeString[1099]): Boolean;
```

(Introduced in version 19)

Parameters for loading are in TRichView.RTFReadProperties [253]. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode [466] and ParaStyleMode [464]. The imported DocX may look absolutely different depending on values of these properties! Setting for DocX loading can be changed in the TRichView component editor [224].

Method type: editing-style method for insertion.

If style templates are used [263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode [260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges a DocX style sheet into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode[462]
)

See also methods of TRichViewEdit:

- InsertDocXFromStreamEd[527].

See also methods of TRichView:

- LoadDocX[326].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417],

See also:

- Saving and loading[127];
- Inserting items at position of caret[119].

6.1.2.2.59 TRichViewEdit.InsertDocXFromStreamEd

Inserts a DocX (Microsoft Word Document) from the **Stream** at the position of the caret.

```
function InsertDocXFromStreamEd(Stream: TStream): Boolean;
```

(Introduced in version 19)

Parameters for loading are in TRichView.RTFReadProperties[253]. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. The imported DocX may look absolutely different depending on values of these properties! Setting for DocX loading can be changed in the TRichView component editor[224].

Method type: editing-style method for insertion.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges a DocX style sheet into Style[260].StyleTemplates[762], and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode[462]
)

See also methods of TRichViewEdit:

- InsertDocXFromFileEd[526].

See also methods of TRichView:

- LoadDocXFromStream[327].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];

- OnStyleTemplatesChange [417].

See also:

- Saving and loading [127];
- Inserting items at position of caret [119].

6.1.2.2.60 TRichViewEdit.InsertHotPicture

Inserts a picture with hypetext link [172] at the position of the caret.

```
function InsertHotPicture(const Name: TRVUnicodeString[1099];
  gr: TRVGraphic[1032]; VAlign: TRVVAlign[1100];
  ImageWidth: TRVStyleLength[1094] = 0;
  ImageHeight: TRVStyleLength[1094] = 0: Boolean;
```

(introduced in version 1.5; changed in version 21)

The same as InsertPicture [534], 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 [274].

See also:

- Inserting items at position of caret [119];
- Hypertext in RichView [97];
- Item types [90].

6.1.2.2.61 TRichViewEdit.InsertHotspot

Inserts a *hotspot* [178] at the position of the caret.

VCL and LCL:

```
function InsertHotspot(ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList): Boolean;
```

FireMonkey:

```
function InsertHotspot(ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList;
  ImageWidth, ImageHeight: TRVStyleLength[1094]): 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.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

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:

- AddHotspot [275].

See also:

- Inserting items at position of caret [119];
- Hypertext in RichView [97];
- Item types [90].

6.1.2.2.62 TRichViewEdit.InsertHTMLFromFileEd

Inserts an HTML from the file **FileName** at the position of the caret.

```
function InsertHTMLFromFileEd(
  const FileName: TRVUnicodeString[1099];
  CodePage: TRVCodePage[1062] = CP_ACP): Boolean;
```

(Introduced in version 21)

Parameters

FileName – the name of the input HTML file.

CodePage defines encoding of HTML file. If it is equal to 0 (CP_ACP), encoding is detected by TRichView.

Method type: editing-style method for insertion.

Parameters for loading are in TRichView.HTMLReadProperties [243].

If style templates are used [263], and StyleTemplateInsertMode [260] <>*rvstimIgnoreSourceStyleTemplates*, the method merges a HTML style sheet into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

See details about HTML style sheet and encoding detection in the topic about LoadHTML [330].

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertHTMLFromStreamEd [530];
- PasteHTML [545].

See also methods of TRichView:

- LoadHTML [330].

See also events of TRichView:

- OnImportPicture [386];
- OnImportFile [385];
- OnReadHyperlink [395];
- OnStyleTemplatesChange [417],

See also:

- Saving and loading [127];
- Inserting items at position of caret [119].

6.1.2.2.63 TRichViewEdit.InsertHTMLFromStreamEd

Inserts an HTML from the stream **Stream** at the position of the caret.

```
function InsertHTMLFromStreamEd(Stream: TStream;
  const Path: TRVUnicodeString = '';
  CodePage: TRVCodePage[1062] = CP_ACP): Boolean;
```

(Introduced in version 21)

Parameters

Stream – stream containing HTML

Path defines a base path for images. If HTMLReadProperties [243].BasePathLinks [433] = *True*, this path is also used for hyperlinks.

CodePage defines encoding of HTML file. If it is equal to 0 (CP_ACP), encoding is detected by TRichView.

Method type: editing-style method for insertion.

Parameters for loading are in TRichView.HTMLReadProperties [243].

If style templates are used [263], and StyleTemplateInsertMode [260] <>*rvstimIgnoreSourceStyleTemplates*, the method merges a HTML style sheet into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

See details about HTML style sheet and encoding detection in the topic about LoadHTML [330].

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertHTMLFromFileEd [529];
- PasteHTML [545].

See also methods of TRichView:

- LoadHTML [330].

See also events of TRichView:

- OnImportPicture [386];
- OnImportFile [385];
- OnReadHyperlink [395];
- OnStyleTemplatesChange [417],

See also:

- Saving and loading [127];
- Inserting items at position of caret [119].

6.1.2.2.64 TRichViewEdit.InsertItem

Inserts an item at the position of the caret.

```
function InsertItem(const Name: TRVUnicodeString[1099];
  Item: TCustomRVItemInfo[899]): 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 [179], labels [181], sequences [183], footnotes [185], endnotes [184], references to notes [190], page numbers [190], page counts [192], equations [193] (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 [757]).

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 [488] or protected [815], or when inserting in table having multicell selection); in this case **Item** was destroyed (freed) by InsertItem.

See also methods of TCustomRichView:

- AddItem [276].

See also:

- Inserting items at position of caret [119];
- Item types [90].

6.1.2.2.65 TRichViewEdit.InsertMarkdownFromFileEd

Inserts a ***Markdown*** from file **FileName** at the position of the caret.

```
function InsertMarkdownFromFileEd(const FileName: TRVUnicodeString[1099];
  CodePage: TRVCodePage[1062] = CP_UTF8): Boolean;
```

(Introduced in version 20)

By default, the method assumes that the file has UTF-8 encoding; but you can specify another **CodePage** in the parameter.

Parameters for loading are in MarkdownProperties[245]. See the topic about the class of this property[447] for the detailed explanation of Markdown import and export.

Unlike other file insertion methods, Markdown loading does not have an option to restrict formatting to existing text, paragraphs and list styles: it may add new styles.

By default, hyperlink targets are loaded in items tags[96]. You can customize loading using OnReadHyperlink[395] event.

Method type: editing-style method for insertion.

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertMarkdownFromStreamEd[532].

See also methods of TRichView:

- LoadMarkdown[332].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395].

See also:

- Saving and loading[127];
- Inserting items at position of caret[119].

6.1.2.2.66 TRichViewEdit.InsertMarkdownFromStreamEd

Inserts a ***Markdown*** from stream **Stream** at the position of the caret.

```
function InsertMarkdownFromStreamEd(Stream: TStream;
  const Path: TRVUnicodeString[1099] = '';
  CodePage: TRVCodePage[1062] = CP_UTF8): Boolean;
```

(Introduced in version 20)

By default, the method assumes that the stream has UTF-8 encoding; but you can specify another **CodePage** in the parameter.

Path defines a base path for images. If *rvmdloBasePathLinks* is included MarkdownProperties[245].LoadOptions[453], this path is also used for hyperlinks.

Parameters for loading are in MarkdownProperties [245]. See the topic about the class of this property [447] for the detailed explanation of Markdown import and export.

Unlike other stream insertion methods, Markdown loading does not have an option to restrict formatting to existing text, paragraphs and list styles: it may add new styles.

By default, hyperlink targets are loaded in items tags [96]. You can customize loading using OnReadHyperlink [395] event.

Method type: editing-style method for insertion.

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertMarkdownFromFileEd [532].

See also methods of TRichView:

- LoadMarkdownFromStream [333].

See also events of TRichView:

- OnImportPicture [386];
- OnReadHyperlink [395].

See also:

- Saving and loading [127];
- Inserting items at position of caret [119].

6.1.2.2.67 TRichViewEdit.InsertOEMTextFromFile

Loads a text file (in OEM-defined character set) **FileName** and inserts its content at the position of the caret.

```
function InsertOEMTextFromFile(const FileName: TRVUnicodeString[1099]):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.

Platform note: this method inserts OEM text only in Windows version. For other platforms, it works like InsertTextFromFile [541].

Return value:

"Was reading from from file successful?"

See also methods:

- InsertTextFromFile [541];
- InsertText [540].

See also properties:

- CurParaStyleNo[481];
- CurTextStyleNo[482].

See also properties of TRVStyle:

- TextStyles[764];
- ParaStyles[757];
- SpacesInTab[761].

See also:

- Saving and loading[127];
- Inserting items at position of caret[119].

6.1.2.2.68 TRichViewEdit.InsertPageBreak

Inserts an explicit page break at the position of the 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[325]), the method breaks the current line (just like if user pressed **Enter** key).

The method sets "explicit page break" flag for the item at the position of caret (see PageBreaksBeforeItems[250]).

Method type: editing-style.

See also:

- RemoveCurrentPageBreak[551].

See also properties of RichView:

- PageBreaksBeforeItems[250];
- Options[247] (*rvoShowPageBreaks*).

See also properties of RVStyle:

- PageBreakColor[756].

6.1.2.2.69 TRichViewEdit.InsertPicture

Inserts a picture[169] at the position of the caret.

```
function InsertPicture(const Name: TRVUnicodeString[1099]; gr: TRVGraphic[1032];
  VAlign: TRVVAlign[1100];   ImageWidth: TRVStyleLength[1094] = 0;
  ImageHeight: TRVStyleLength[1094] = 0): Boolean;
```

(changed in version 18 and 21)

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[1067], 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[1114]). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign[1100] for possible values.

The image is displayed stretched to **ImageWidth** x **ImageHeight** (if they are non-zero, see *rvepImageWidth* and *rvepImageHeight* in IDH_Type_TRVExtraItemProperty[1067]).

Method type: editing-style method for insertion.

If you want this picture to be a hypertext link, use InsertHotPicture[528] 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: TRVGraphic[1032];
...
if OpenPictureDialog1.Execute then
  gr := RVGraphicHandler[1125].LoadFromFile[1033](OpenPictureDialog1.FileName);
  if gr<>nil then
    MyRichViewEdit.InsertPicture('', gr, rvvaBaseline)
  else
    // process a picture loading error here
end;
```

See also methods of TCustomRichView:

- AddPicture[278].

See also:

- Inserting items at position of caret[119];
- Hypertext in RichView[97];
- Item types[90].

6.1.2.2.70 TRichViewEdit.InsertRTFFromFileEd

Inserts an RTF (Rich Text Format) from file **FileName** at the position of the caret.

```
function InsertRTFFromFileEd(const FileName: TRVUnicodeString[1099]): Boolean;
```

(Introduced in version 1.5; changed in version 18)

Parameters for loading are in TRichView.RTFReadProperties[253]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. 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[224].

Method type: editing-style method for insertion.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style[260].StyleTemplates[762], and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.RTFErrorCode[466])

See also methods of TRichViewEdit:

- InsertRTFFromStreamEd[536];
- PasteRTF[547].

See also methods of TRichView:

- LoadRTF[334].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417],

See also:

- Saving and loading[127];
- Inserting items at position of caret[119].

6.1.2.2.71 TRichViewEdit.InsertRTFFromStreamEd

Inserts an RTF (Rich Text Format) from the **Stream** at the position of the caret.

```
function InsertRTFFromStreamEd(Stream: TStream): Boolean;
```

(Introduced in version 1.5)

Parameters for loading are in TRichView.RTFReadProperties[253]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[466] and ParaStyleMode[464]. 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[224].

Method type: editing-style method for insertion.

If style templates are used[263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style[260].StyleTemplates[762], and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.RTFErrorCode[466])

See also methods of TRichViewEdit:

- InsertRTFFromFileEd[536];
- PasteRTF[547].

See also methods of TRichView:

- LoadRTFFromStream[335].

See also events of TRichView:

- OnImportPicture[386];
- OnReadHyperlink[395];
- OnStyleTemplatesChange[417].

See also:

- Saving and loading[127];
- Inserting items at position of caret[119].

6.1.2.2.72 TRichViewEdit.InsertRVFFromFileEd

Inserts an RVF (RichView Format[129]) from the file **FileName** at the position of the caret.

```
function InsertRVFFromFileEd(const FileName: TRVUnicodeString[1099]):Boolean;
```

(changed in version 18)

Settings for RVF loading can be changed in the TRichView component editor[224].

Method type: editing-style method for insertion.

If style templates are used[263], and RVFTextStylesReadMode[258]=RVFParaStylesReadMode[258] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style[260].StyleTemplates[762], and reads text and paragraph styles according to StyleTemplateInsertMode[260]. The method calls OnStyleTemplatesChange[417] event.

Return value: "successful insertion?"

See also methods of TRichViewEdit:

- InsertRVFFromStreamEd[538];
- PasteRVF[547].

See also methods of TRichView:

- LoadRVF[336].

- **See also properties:**
- RVFOptions [254];
- RVFWarnings [258];
- RVFTextStylesReadMode [258] (this method type: RVF insertion);
- RVFParaStylesReadMode [258] (this method type: RVF insertion);
- UseStyleTemplates [263];
- StyleTemplateInsertMode [260].

See also events:

- OnRVFImageListNeeded [400];
- OnRVFControlNeeded [400];
- OnRVFPictureNeeded [401];
- OnControlAction [380] (ControlAction=*rvcaAfterRVFLoad*).

See also:

- TRichView component editor [224];
- Saving and loading [127];
- Inserting items at position of caret [119];
- RVF [129].

6.1.2.2.73 TRichViewEdit.InsertRVFFromStreamEd

Inserts an RVF (RichView Format [129]) from the **Stream** at the position of the caret.

```
function InsertRVFFromStreamEd(Stream: TStream):Boolean;
```

Settings for RVF loading can be changed in the TRichView component editor [224].

Method type: editing-style method for insertion.

If style templates are used [263], and RVFTextStylesReadMode [258]=RVFParaStylesReadMode [258] =*rvf_sInsertMerge*, and StyleTemplateInsertMode [260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

Return value: "successful insertion?"

See also methods of TRichViewEdit:

- InsertRVFFromFileEd [537];
- PasteRVF [547].

See also methods of TRichView:

- LoadRVFFromStream [335];
- InsertRVFFromStream [323];
- AppendRVFFromStream [283].
- **See also properties:**
- RVFOptions [254];
- RVFWarnings [258];
- RVFTextStylesReadMode [258] (this method type: RVF insertion);
- RVFParaStylesReadMode [258] (this method type: RVF insertion);
- UseStyleTemplates [263];

- StyleTemplateInsertMode [260].

See also events:

- OnRVFImageListNeeded [400];
- OnRVFControlNeeded [400];
- OnRVFPictureNeeded [401];
- OnControlAction [380] (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange [417].

See also:

- TRichView component editor [224];
- Saving and loading [127];
- Inserting items at position of caret [119];
- RVF [129].

6.1.2.2.74 TRichViewEdit.InsertStringTag -A -W

The methods insert a text string **s** with an associated **Tag** at the position of the caret.

```
function InsertStringTag(const s: String;
  const Tag: TRVTag[1096]): Boolean;
function InsertStringATag(const s: TRVAnsiString[1056];
  const Tag: TRVTag[1096]): Boolean;
function InsertStringWTag(const s: TRVUnicodeString[1099];
  const Tag: TRVTag[1096]): Boolean;
```

(introduced in version 1.6)

Inserted text (one text item) has the current text [482] and current paragraph [481] style.

Methods type: editing-style method for insertion.

Unlike InsertText [540]:

- **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 [761]=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 [540] when possible.

Unicode note:

Internally, text is stored as Unicode. **InsertStringATag** converts ANSI to Unicode using Style [260].DefCodePage [745].

InsertStringTag works:

- like **InsertStringATag**, in Delphi 2007 and older
- like **InsertStringWTag**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus [160]

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[540].

See also properties:

- CurParaStyleNo[481];
- CurTextStyleNo[482].

See also properties of TRVStyle:

- TextStyles[764];
- ParaStyles[757];
- SpacesInTab[761].

See also:

- Inserting items at position of caret[119];
- Tags[96].

6.1.2.2.75 TRichViewEdit.InsertTab

Inserts a tabulator[168] at the position of the caret.

```
function InsertTab: Boolean;
```

(introduced in v1.9)

Inserted tabulator has the current text[482] and the current paragraph[481] style.

This method always inserts a tabulator, regardless of RVStyle.SpacesInTab[761] mode.

If RVStyle.SpacesInTab[761]=0, this method does the same work as InsertText[540](#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[280].

6.1.2.2.76 TRichViewEdit.InsertText -A -W

The methods insert a **text** at the position of the caret.

```
procedure InsertText(const text: String;
  CaretBefore: Boolean=False);
procedure InsertTextA(const text: TRVAnsiString[1056];
  CaretBefore: Boolean=False);
procedure InsertTextW(const text: TRVUnicodeString[1099];
  CaretBefore: Boolean=False);
```

Inserted text has current text[482] and current paragraph[481] style.

Text may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[168], depending on value of SpacesInTab[761] property of the linked[260] RVStyle

component. #12 characters add page breaks[250]. 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** converts ANSI to Unicode using Style[260].DefCodePage[745].

InsertText works:

- like **InsertTextA**, in Delphi 2007 and older
- like **InsertTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus[160]

See also methods:

- InsertStringTag, InsertStringATag, InsertStringWTag[539].

See also properties:

- CurParaStyleNo[481];
- CurTextStyleNo[482].

See also properties of TRVStyle:

- TextStyles[764];
- ParaStyles[757];
- SpacesInTab[761].

See also:

- Inserting items at position of caret[119].

6.1.2.2.77 TRichViewEdit.InsertTextFromFile -W

The methods load a text file **FileName** and insert its content at the position of the caret. **InsertTextFromFile** inserts text from ANSI text files, **InsertTextFromFileW** inserts text from Unicode (UTF-16) text files.

```
function InsertTextFromFile(const FileName: TRVUnicodeString[1099];
  CodePage: Cardinal=CP_ACP): Boolean;
function InsertTextFromFileW(const FileName: TRVUnicodeString[1099]):
  Boolean;
```

Inserted text has current text[482] and current paragraph[481] style.

A file may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[168], depending on value of SpacesInTab[761] property of the linked[260] RVStyle

component. #12 characters add page breaks [250]. 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 [533];
- InsertText, -A, -W [540].

See also methods of TRichView:

- LoadText, LoadTextW [338].

See also properties:

- CurParaStyleNo [481];
- CurTextStyleNo [482].

See also properties of TRVStyle:

- TextStyles [764];
- ParaStyles [757];
- SpacesInTab [761].

See also:

- Saving and loading [127];
- Inserting items at position of caret [119].

6.1.2.2.78 TRichViewEdit.MoveCaret

Moves the caret in the specified direction.

```
type
  TRVCaretMovement = (rvcmUp, rvcmDown,
    rvcmLeft, rvcmRight, rvcmTop, rvcmBottom,
    rvcmHome, rvcmEnd, rvcmParaBeginning,
    rvcmNextParaBeginning);
procedure MoveCaret(Direction: TRVCaretMovement;
  MakeSelection: Boolean = False);
```

(introduced in version 20)

The editor must be formatted before calling this method.

Direction	Moves the caret...
rvcmUp	up

Direction	Moves the caret...
rvcmDown	down
rvcmLeft	to the left
rvcmRight	to the right
rvcmTop	to the beginning of the document
rvcmBottom	to the end of the document
rvcmHome	to the beginning of the current line
rvcmEnd	to the end of the current line
rvcmParaBeginning	to the beginning of the current paragraph
rvcmNextParaBeginning	to the beginning of the next paragraph

If **MakeSelection** = *True*, a content from the old caret position to the new caret position is selected.

6.1.2.2.79 TRichViewEdit.Paste

Pastes document from the Clipboard.

```
procedure Paste;
```

First, it generates OnPaste[594] 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[478] property into account: it tries to paste the specific format only if it is included in the property.

The method tries to paste as RVF, see PasteRVF[547].

If unsuccessful, it tries to paste as RTF, see PasteRTF[547].

If unsuccessful, it tries to paste as HTML, see PasteRTF[545].

If unsuccessful, it tries to paste as an URL, see PasteURL[549].

If unsuccessful, it tries to paste as text, see PasteTextW[548].

If unsuccessful, it tries to paste image files, see PasteGraphicFile[544] (if *rvoAssignImageFileNames* in Options[247], file name is is stored in *rvespImageFileName* extra string property[1072]; you can modify this file name in OnAssignImageFileName[379] event).

If unsuccessful, it tries to paste as a bitmap, see PasteBitmap[544].

If unsuccessful, it tries to paste as a metafile picture, see PasteMetafile[546].

Thie method never attempts to paste Markdown. Use PasteMarkdown[546] instead.

This method is executed automatically when user presses Ctrl+V (Command+V on macOS) or Shift+Insert.

Method type: editing-style method for insertion.

See also methods:

- CanPaste [507].

See also:

- TRichView and Clipboard [113].

6.1.2.2.80 TRichViewEdit.PasteBitmap

Inserts Windows bitmap from the Clipboard at 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.

Supported platforms: Windows (VCL, LCL, FireMonkey), macOS (FireMonkey)

Return value:

True, if an image was inserted.

See also methods:

- Paste [543];
- InsertPicture [534].

See also properties:

- DefaultPictureVAlign [483].

See also:

- RichView and Clipboard [113].

6.1.2.2.81 TRichViewEdit.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 [1072]. You can modify this file name in OnAssignImageFileName [379] event.

Method type: editing-style method for insertion.

You can use OnDropFile [586] 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 [247]).

Supported platforms: Windows (VCL, LCL, FireMonkey)

Return value:

True, if at least one of pictures was inserted.

See also methods:

- Paste [543];
- InsertPicture [534].

See also properties:

- DefaultPictureVAlign [483].

See also events:

- OnAssignImageFileName [379]

See also:

- RichView and Clipboard [113].

6.1.2.2.82 TRichViewEdit.PasteHTML

Inserts HTML from the Clipboard at the position of the caret.

```
function PasteHTML: Boolean;
```

(Introduced in version 21)

This method does nothing if the Clipboard does not contain HTML.

Parameters for loading are in TRichView.HTMLReadProperties [243].

Method type: editing-style method for insertion.

If style templates are used [263], and StyleTemplateInsertMode [260] <>*rvstimIgnoreSourceStyleTemplates*, the method merges a HTML style sheet into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. However, most browsers copy HTML using inline CSS attributes, so StyleTemplateInsertMode [260] works not as good as for insertion of HTML files or streams where a style sheet is defined in a header.

The method calls OnStyleTemplatesChange [417] event.

See the details about HTML style sheet in the topic about LoadHTML [330].

Platform notes: this method is implemented for Windows, macOS, and Linux. For other platforms, it does nothing.

Return value:

True if HTML was successfully pasted

See also methods:

- CanPasteHTML[507];
- InsertHTMLFromFileEd[529];
- InsertHTMLFromStreamEd[530].

See also:

- TRichView and Clipboard[113]

6.1.2.2.83 TRichViewEdit.PasteMarkdown

Inserts text from the Clipboard at the position of the caret, processing it as Markdown.

```
function PasteMarkdown: Boolean;
```

(Introduced in version 24)

This method does nothing if the Clipboard does not contain a plain text.

Parameters for reading Markdown are in TRichView.MarkdownProperties[245].

This method is an alternative to PasteText[548]: the both methods paste the same data (a plain text), but this method processes pasted text as Markdown.

Method type: editing-style method for insertion.

Return value:

True if Markdown was successfully pasted

See also methods:

- CopyMarkdown[289];
- InsertMarkdownFromFileEd[532];
- InsertMarkdownFromStreamEd[532].

See also:

- TRichView and Clipboard[113]

6.1.2.2.84 TRichViewEdit.PasteMetafile

Inserts metafile from the Clipboard at 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.

Return value:

True, if an image was inserted.

See also methods:

- Paste[543];
- InsertPicture[534].

See also properties:

- DefaultPictureVAlign[483].

See also:

- RichView and Clipboard [113].

6.1.2.2.85 TRichViewEdit.PasteRTF

Inserts RTF (Rich Text Format) from the Clipboard at the position of the caret.

```
function PasteRTF: Boolean;
```

(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.RTFErrorCode [466] property.

Parameters for loading are in TRichView.RTFReadProperties [253]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode [466] and ParaStyleMode [464]. 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 [224].

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 [263], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode [260]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style [260].StyleTemplates [762], and reads text and paragraph styles according to StyleTemplateInsertMode [260]. The method calls OnStyleTemplatesChange [417] event.

Return value:

True if RTF was successfully pasted

See also methods:

- CanPasteRTF [507];
- InsertRTFFromFileEd [536];
- InsertRTFFromStreamEd [536].

See also:

- TRichView and Clipboard [113]

6.1.2.2.86 TRichViewEdit.PasteRVF

Inserts RVF (RichView Format) from the Clipboard at the position of caret

```
function PasteRVF: Boolean;
```

This method does nothing if the Clipboard does not contain RVF.

Method type: editing-style method for insertion.

Return value:

True if RVF was successfully pasted

See also methods:

- CanPasteRVF[507];
- Paste[543];
- InsertRVFFromStreamEd[538].

See also:

- TRichView and Clipboard[113];
- RVF[129].

6.1.2.2.87 TRichViewEdit.PasteText -A -W

These methods Insert text from the Clipboard at the position of caret

```
procedure PasteText;
procedure PasteTextA;
procedure PasteTextW;
```

For Windows: **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 at the position of caret, using the current text[482] and paragraph[481] styles.

Text from the Clipboard may contain special characters: CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[168], depending on value of SpacesInTab[761] property of the linked[260] RVStyle component. #12 characters add page breaks[250]. 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).

Alternatively, you can use PasteMarkdown[546] to insert text as Markdown.

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).

Return value:

True if text was successfully pasted

See also methods:

- Paste[543];
- InsertText, -A, -W[540].

See also properties:

- CurParaStyleNo[481];
- CurTextStyleNo[482].

See also properties of TRVStyle:

- TextStyles[764];
- ParaStyles[757];
- SpacesInTab[761].

See also:

- RichView and Clipboard [113].

6.1.2.2.88 TRichViewEdit.PasteURL

Inserts URL from the Clipboard at 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 [395] 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 [96]).

Supported platforms: VCL and LCL (Windows), FireMonkey (Windows and Android)

Method type: editing-style method for insertion.

Supported platforms: Windows (VCL, LCL, FireMonkey), Android (FireMonkey)

See also methods:

- Paste [543].

See also:

- TRichView and Clipboard [113].

6.1.2.2.89 TRichViewEdit.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 [549] and RedoName [550].

This method must be called only when the document is formatted.

This method is called when the user presses Shift + Ctrl + Z (Shift + Command + Z on macOS) or Shift + Alt + Backspace.

Method type: editing-style.

See also:

- Undo [577];
- Undo in RichViewEdit [120].

6.1.2.2.90 TRichViewEdit.RedoAction

Returns type of the operation to be redone [549] next.

```
function RedoAction: TRVUndoType [1098];
```

(Introduced in version 1.3)

See TRVUndoType[1098] for the list of operations. If the returned value is *rvutCustom*, you can use RedoName[550].

See also:

- UndoAction[577];
- Undo in RichViewEdit[120].

6.1.2.2.91 TRichViewEdit.RedoName

Returns name of the operation to be redone[549] next.

```
function RedoName: TRVUnicodeString[1099];
```

(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[549]).

Returns non-empty string only for custom redo action, initiated by BeginUndoCustomGroup[504].

See also:

- UndoName[578];
- Undo in RichViewEdit[120].

6.1.2.2.92 TRichViewEdit.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[512].

This method is used in conjunction with InsertCheckpoint[525] and GetCheckpointAtCaret[512]. The alternative set of methods is: SetCurrentCheckpointInfo[563], GetCurrentCheckpoint[514] and RemoveCurrentCheckpoint[551].

Method type: editing-style, unlike RemoveCheckpoint[340].

See also:

- "Checkpoints"[92].

6.1.2.2.93 TRichViewEdit.RemoveCheckpointEd

Editing-style method for deleting *checkpoint* associated with the **ItemNo**-th item.

```
procedure RemoveCheckpointEd(ItemNo: Integer);
```

Method type: editing-style, unlike RemoveCheckpoint[340].

See also methods of TRichViewEdit:

- RemoveCurrentCheckpoint[551];
- SetCheckpointInfoEd[560].

See also methods of TRichView:

- GetItemCheckpoint[305];
- RemoveCheckpoint[340].

See also:

- Modifying RichView items[117];
- "Checkpoints"[92].

6.1.2.2.94 TRichViewEdit.RemoveCurrentCheckpoint

Deletes *checkpoint* associated with the item at the position of caret.

```
procedure RemoveCurrentCheckpoint;
```

RemoveCurrentCheckpoint is equivalent to TopLevelEditor[489].RemoveCheckpointEd[550] (TopLevelEditor[489].CurItemNo[480]).

Method type: editing-style.

See also methods:

- GetCurrentCheckpoint[514];
- SetCurrentCheckpointInfo[563];
- RemoveCheckpointEd[550].

See also properties:

- CurItemNo[480].

See also:

- Modifying RichView items[117];
- "Checkpoints"[92].

6.1.2.2.95 TRichViewEdit.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[250]), if this flag was set.

Method type: editing-style.

See also:

- InsertPageBreak[534].

See also properties of RichView:

- PageBreaksBeforeItems[250];
- Options[247] (*rvoShowPageBreaks*).

See also properties of RVStyle:

- PageBreakColor[756].

6.1.2.2.96 TRichViewEdit.RemoveLists

Removes markers [180] (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 [498];
- ChangeListLevels [508].

See also methods of TCustomRichView:

- SetListMarkerInfo [369];
- GetListMarkerInfo [315].

6.1.2.2.97 TRichViewEdit.ResizeControl

Resizes control [174] in the **ItemNo**-th item, and quickly reformats the affected part of the document.

```
procedure ResizeControl(ItemNo, NewWidth, NewHeight: TRVCoord[1064]);
```

The **ItemNo**-th item must be of "inserted control" [174] type (*rvsComponent* [1127]), otherwise the method raises ERichViewError [1014] exception.

The control's width will be set to **NewWidth**, height to **NewHeight**.

Method type: editing-style.

See also methods:

- ResizeCurrentControl [552];
- SetControlInfoEd [560];
- AdjustControlPlacement [497];
- GetItemStyle [310].

See also properties:

- ItemCount [244].

See also:

- Modifying RichView items [117];
- Item types [90].

6.1.2.2.98 TRichViewEdit.ResizeCurrentControl

Resizes control [174] at the position of caret and quickly reformats the affected part of the document.

```
procedure ResizeCurrentControl(NewWidth, NewHeight: TRVCoord[1064]);
```

(introduced in version 1.4)

ResizeCurrentControl(...) is equivalent to TopLevelEditor[489].ResizeControl[552](TopLevelEditor[489].CurItemNo[480], ...).

Method type: editing-style.

See also methods:

- ResizeControl[552];
- SetControlInfoEd[560];
- AdjustControlPlacement[497];
- GetItemStyle[310].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- Item types[90].

6.1.2.2.99 TRichViewEdit.SearchText -W

The methods search for the substring **s** in the document.

```
function SearchText(s: String;
  SrchOptions: TRVESearchOptions[1066]): Boolean;
function SearchTextA(s: TRVAnsiString[1056];
  SrchOptions: TRVESearchOptions[1066]): Boolean;
function SearchTextW(s: TRVUnicodeString[1099];
  SrchOptions: TRVESearchOptions[1066]): Boolean;
```

When found, these methods select substring and move the caret to the end of selection (this behavior can be modified in OnTextFound[418] event, see below).

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 *rvseoFromStored* is included in **SrchOptions**, the last stored search position[375] if used instead of selection or a caret position.

If *rvseoMultiItem* is included, the search can match substrings of several text items[167]. 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** converts **s** to Unicode using (Style[260].DefCodePage[745]) and calls **SearchTextW**.

SearchText works:

- like **SearchTextA**, in Delphi 2007 and older
- like **SearchTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus [160]

These methods can be used for search-and-replace operation as well. **SearchText** selects the found string, InsertText [540] replaces it with the new one. See important note about search and replace in down-up direction in the topic about InsertText [540].

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.

When the text is found, OnTextFound [418] event is called. If not disallowed in this event, the found text is selected. A selection requires a formatted document, so (unless you do your own processing in OnTextFound [418] and disallow a selection) these methods must be called only when the document is formatted.

Return value:

Is the substring found?

See also:

- TRichView.SearchText, -A, W [354];
- GetRVESearchOptions [1049] global function;
- Searching in TRichView and TRichViewEdit [115].

6.1.2.2.100 TRichViewEdit.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 [416] event.

This method must be called only when the document is formatted.

See also methods:

- SelectCurrentWord [555];
- GetSelText [321].

See also:

- Selecting in RichView document [112].

6.1.2.2.101 TRichViewEdit.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[416] event.

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[235] property.

This method must be called only when the document is formatted.

See also methods:

- SelectWordAt[356];
- GetSelText[321];
- SelectCurrentLine[554].

See also:

- Selecting in RichView document[112].

6.1.2.2.102 TRichViewEdit.Set*PropertyEd

Editing-style methods for changing properties of the editor.

```
procedure SetIntPropertyEd(Prop: TRVIntProperty[1080];
  Value: Integer;  AutoReformat: Boolean);
procedure SetStrPropertyEd(Prop: TRVStrProperty[1093];
  const Value: TRVUnicodeString[1099]; AutoReformat: Boolean);
procedure SetFloatPropertyEd(Prop: TRVFloatProperty[1074];
  Value: TRVLength[1081]; 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[508] 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(505)(rvutLayout);
  MyRichViewEdit.SetUndoGroupMode(577)(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(577)(False);
  MyRichViewEdit.Reformat(339);
  MyRichViewEdit.Change(508);
end;
```

See also methods:

- SetBackgroundImageEd(556);
- SetBackgroundPropertyEd(557).

6.1.2.2.103 TRichViewEdit.SetBackgroundImageEd

An editing-style method for changing BackgroundPicture(229).

```
procedure SetBackgroundImageEd(Graphic: TRVGraphic(1032);
  AutoReformat: Boolean);
```

(introduced in version 15)

Parameters:

Graphic – a graphic to assign to BackgroundPicture(229). 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(508) 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 BackgroundPicture(229)).

Example:

```
if MyRichViewEdit.CanChange then
begin
  MyRichViewEdit.BeginUndoGroup(505)(rvutBackground);
  MyRichViewEdit.SetUndoGroupMode(577)(True);
  MyRichViewEdit.SetBackgroundPropertyEd(557)(rvbpidVisible, Ord(True),
    False);
  MyRichViewEdit.SetBackgroundImageEd(Image1.Picture.Graphic, False);
```

```
  MyRichViewEdit.SetUndoGroupMode(False);
  MyRichViewEdit.Invalidate;
  MyRichViewEdit.Change;
end;
```

See also methods:

- SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd.

6.1.2.2.104 TRichViewEdit.SetBackgroundPropertyEd

An editing-style method for changing BackgroundProperties.

```
procedure SetBackgroundPropertyEd(Prop: TRVBackgroundPropertyId;
  Value: Integer; AutoReformat: Boolean);
```

(introduced in version 24)

Parameters:

Prop identifies a property of BackgroundProperties.

Prop – new value for this property.

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 sub-properties of BackgroundProperties).

Example:

```
if MyRichViewEdit.CanChange then
begin
  MyRichViewEdit.BeginUndoGroup(rvutBackground);
  MyRichViewEdit.SetUndoGroupMode(True);
  MyRichViewEdit.SetBackgroundPropertyEd(rvbpidVisible, Ord(True),
    False);
  MyRichViewEdit.SetBackgroundImageEd(Image1.Picture.Graphic, False);
  MyRichViewEdit.SetUndoGroupMode(False);
  MyRichViewEdit.Invalidate;
  MyRichViewEdit.Change;
end;
```

See also methods:

- SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd.

6.1.2.2.105 TRichViewEdit.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: TRVColor; const ATag: TRVTag);
```

Parameters:

ItemNo – index of the item. The item must be of *break*[173] type (*rvsBreak*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item.

Items of subdocuments (table cells[951]) 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[372]), and call SetBreakInfoEd of this inplace-editor. Alternatively, you can use SetCurrentBreakInfo[561].

AWidth – line width (or rectangle height). This value is measured in Style[260].Units[764].

AStyle – visual style of this *break*, see TRVBreakStyle[1059] for possible values.

AColor – line color. If it is equal to *rvclNone*[1105], Style[260].TextStyles[764][0].Color[810] is used.

ATag – *tag* of the item. You can use value returned by GetBreakInfo[296] or GetItemTag[310] for this item.

Instead of this method, you can use SetItemExtraIntPropertyExEd[572] and SetItemTagEd[574] methods.

Method type: editing-style (unlike SetBreakInfo[358]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[572] and SetItemExtraStrPropertyEd[573].

When possible, use SetCurrentBreakInfo[561] instead of this method.

See also methods of TCustomRichView:
- GetBreakInfo[296];
- GetItemStyle[310];
- SetBreakInfo[358].

See also methods:
- SetCurrentBreakInfo[561].

See also properties:
- ItemCount[244].

See also:
- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.106 TRichViewEdit.SetBulletInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of TRichView, if this item is a *bullet*[176] *or hotspot*[178].

VCL and LCL:

```
procedure SetBulletInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString[1099]; AImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag[1096]);
```

FireMonkey:

```
procedure SetBulletInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString[1099]; AImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag[1096];
  ImageWidth, ImageHeight: TRVStyleLength[1094]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *bullet*[176] or *hotspot*[178] type (*rvsBullet or rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item.

Items of subdocuments (table cells[951]) 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[372]), and call SetBulletInfoEd of this inplace-editor. Alternatively, you can use SetCurrentBulletInfo[562].

AName – name of *bullet*[176], any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW[574] method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyExEd[572] method.

ATag – *tag* of the item. You can use value returned by GetBulletInfo[297] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTagEd[574] method.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Method type: editing-style (unlike SetBulletInfo[359]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[572] and SetItemExtraStrPropertyEd[573].

When possible, use SetCurrentBulletInfo[562] instead of this method.

See also methods of TCustomRichView:

- GetBulletInfo[297];
- GetItemStyle[310];
- SetBulletInfo[359].

See also methods:

- SetCurrentBulletInfo[562].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.107 TRichViewEdit.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[1096];
  const AName: TRVUnicodeString[1099]; ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to ItemCount[244]-1. Items of subdocuments (table cells[951]) 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[372]), and call SetCheckpointInfoEd of this inplace-editor. Alternatively, you can use SetCurrentCheckpointInfo[563].

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[379] event for this *checkpoint*.

Method type: editing-style (unlike SetCheckpointInfo[360]).

When possible, use SetCurrentCheckpointInfo[563] instead of this method.

See also methods of TCustomRichView:

- GetItemCheckpoint[305];
- SetCheckpointInfo[360].

See also methods:

- SetCurrentCheckpointInfo[563].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- "Checkpoints"[92];
- "Tags"[96].

6.1.2.2.108 TRichViewEdit.SetControlInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is "inserted control"[174].

```
procedure SetControlInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString[1099]; AVAlign: TRVVAlign[1100];
  const ATag: TRVTag[1096]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control 174 type (*rvsComponent* 1127), otherwise the method raises ERichViewError 1014 exception. Items are indexed from 0 to ItemCount 244-1, GetItemStyle 310 returns type of item.

Items of subdocuments (table cells 951) 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 372), and call SetControlInfoEd of this inplace-editor. Alternatively, you can use SetCurrentControlInfo 564.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW 574 method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign 1100 for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetControlInfo 301 or GetItemTag 310 for this item. The *tag* can also be set by SetItemTagEd 574 method.

Method type: editing-style (unlike SetControlInfo 361).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd 572 and SetItemExtraStrPropertyEd 573.

When possible, use SetCurrentControlInfo 564 instead of this method.

See also methods of TCustomRichView:

- GetControlInfo 301;
- GetItemStyle 310;
- SetControlInfo 361.

See also methods:

- SetCurrentControlInfo 564;
- ResizeControl 552;
- AdjustControlPlacement 497.

See also properties:

- ItemCount 244.

See also:

- Modifying RichView items 117;
- Item types 90;
- "Tags" 96.

6.1.2.2.109 TRichViewEdit.SetCurrentBreakInfo

Changes properties for the item at the position of caret, if this item is a *break* 173

```
procedure SetCurrentBreakInfo(AWidth: TRVStyleLength1094;
  AStyle: TRVBreakStyle1059;  AColor: TRVColor1063; const ATag: TRVTag1096);
```

SetCurrentBreakInfo(...) is equivalent to TopLevelEditor 489.SetBreakInfoEd 557(TopLevelEditor 489.CurItemNo 480, ...).

This method can be used if CurItemStyle 481 returns *rvsBreak* 1127.

Parameters:

AWidth – line width (or rectangle height). This value is measured in Style[260].Units[764].

AStyle – visual style of this *break*, see TRVBreakStyle[1059] for possible values.

AColor – line color. If it is equal to *rvclNone*[1105], Style[260].TextStyles[764][0].Color[810] is used.

ATag – *tag* of the item. You can use value returned by GetCurrentBreakInfo[512] or GetCurrentTag[522] for this item.

Instead of this method, you can use SetCurrentItemExtraIntPropertyEx[566] and SetCurrentTag[570] methods.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[566] and SetCurrentItemExtraStrProperty[567].

See also methods:

- GetCurrentBreakInfo[512];
- SetBreakInfoEd[557].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.110 TRichViewEdit.SetCurrentBulletInfo

Changes properties for the item at the position of caret, if this item is a *bullet*[176] *or hotspot*[178].

VCL and LCL:

```
procedure SetCurrentBulletInfo(const AName: TRVUnicodeString[1099];
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure SetCurrentBulletInfo(const AName: TRVUnicodeString[1099];
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096];
  AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

SetCurrentBulletInfo(...) is equivalent to TopLevelEditor[489].**SetBulletInfoEd**[558](TopLevelEditor[489].CurItemNo[480], ...).

This method can be used if CurItemStyle[481] returns *rvsBullet or rvsHotspot*[1127].

Parameters:

AName – name of *bullet*[176], any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW[568] method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetCurrentItemExtraIntPropertyEx[566] method.

ATag – *tag* of the item. You can use value returned by GetCurrentBulletInfo[513] or GetCurrentTag[522] for this item. The *tag* can also be set by SetCurrentTag[570] method.

Additional parameters for FireMonkey version:

AImageWidth, AImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[566] and SetCurrentItemExtraStrProperty[567].

See also methods:

- GetCurrentBulletInfo[513];
- SetBulletInfoEd[558].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.111 TRichViewEdit.SetCurrentCheckpointInfo

Creates or modifies *checkpoint* associated with the item at the position of caret.

```
function SetCurrentCheckpointInfo(const ATag: TRVTag[1096];
  const AName: TRVUnicodeString[1099]; ARaiseEvent: Boolean): Boolean;
```

(changed in version 18 and 19)

SetCurrentCheckpointInfo(...) is equivalent to TopLevelEditor[489].**SetCheckpointInfoEd**[560] (TopLevelEditor[489].CurItemNo[480], ...).

If the item at the position of caret already has a *checkpoint* (see GetCurrentCheckpoint[514]), its properties are modified; otherwise, a new *checkpoint* is created.

This method is used in conjunction with GetCurrentCheckpoint[514] and RemoveCurrentCheckpoint[551]. The alternative set of methods is: InsertCheckpoint[525], GetCheckpointAtCaret[512] and RemoveCheckpointAtCaret[550].

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[379] event for this *checkpoint*.

Return value:

True if the checkpoint was set. *False* otherwise (the function may fail because of a protection or a read-only mode).

Method type: editing-style.

See also methods:

- GetCurrentCheckpoint[514], RemoveCurrentCheckpoint[551];
- InsertCheckpoint[525], GetCheckpointAtCaret[512], RemoveCheckpointAtCaret[550];
- SetCheckpointInfoEd[560].

See also properties:

- CurItemNo[480].

See also:

- Modifying RichView items[117];
- "Checkpoints"[92];
- "Tags"[96].

6.1.2.2.112 TRichViewEdit.SetCurrentControlInfo

Modifies the item at the position of caret, if this item is an inserted control[174].

```
procedure SetCurrentControlInfo(const AName: TRVUnicodeString[1099];
  AVAlign: TRVVAlign[1100]; const ATag: TRVTag[1096]);
```

(changed in version 18)

SetCurrentControlInfo(...) is equivalent to TopLevelEditor[489].**SetControlInfoEd**[560] (TopLevelEditor[489].CurItemNo[480], ...).

This method can be used if CurItemStyle[481] returns *rvsComponent*[1127].

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemText[568] method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign[1100] for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetCurrentControlInfo[515] or GetCurrentTag[522] for this item. The *tag* can also be set by SetCurrentTag[570] method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[566] and SetCurrentItemExtraStrProperty[567].

See also methods:

- GetCurrentControlInfo[515];
- SetControlInfoEd[560].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.113 TRichViewEdit.SetCurrentHotspotInfo

Modifies the item at the position of caret, if this item is a *hotspot*[178].

VCL and LCL:

```
procedure SetCurrentHotspotInfo(const AName: TRVUnicodeString[1099];
  AImageIndex, AHotImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096]);
```

FireMonkey:

```
procedure SetCurrentHotspotInfo(const AName: TRVUnicodeString[1099];
  AImageIndex, AHotImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[1096];
  AImageWidth, AImageHeight: TRVStyleLength[1094]);
```

(changed in version 18)

SetCurrentHotspotInfo(...) is equivalent to TopLevelEditor[489].**SetHotspotInfoEd**[570] (TopLevelEditor[489].CurItemNo[480], ...).

This method can be used if CurItemStyle[481] returns *rvsHotspot*[1127].

Parameters:

AName – name of *hotspot*[178], any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW[568] method.

AImageList – not used, reserved, set it to *nil*..

AImageIndex – index of image in the image list. It can also be set using SetCurrentItemExtraIntPropertyEx[566] 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[566] method.

ATag – *tag* of the item. You can use value returned by GetCurrentHotspotInfo[515] or GetCurrentTag[522] for this item. The *tag* can also be set by SetCurrentTag[570] method.

Additional parameters for FireMonkey version:

AImageWidth, AImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[566] and SetCurrentItemExtraStrProperty[567].

See also methods:

- GetCurrentHotspotInfo[515];
- SetHotspotInfoEd[570].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.114 TRichViewEdit.SetCurrentItemExtraIntProperty -Ex

The methods change value of an integer property for the item at the position of caret.

```
procedure SetCurrentItemExtraIntProperty(Prop: TRVExtraItemProperty[1067];
  Value: Integer; AutoReformat: Boolean);
procedure SetCurrentItemExtraIntPropertyEx(Prop, Value: Integer;
  AutoReformat: Boolean);
```

(introduced in versions 1.7 and 15)

SetCurrentItemExtraIntProperty[Ex](...) is equivalent to TopLevelEditor[489].**SetItemExtraIntProperty[Ex]Ed**[572](TopLevelEditor[489].CurItemNo[480], ...).

These methods set a new **Value** of the item's property identified by **Prop**.

In **SetCurrentItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[1067]. 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[1119]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[582].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[508] 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[503] and EndItemModify[511]);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call Reformat[339]).

Method type: editing-style.

See also methods:

- GetCurrentItemExtraIntProperty[Ex][518];
- SetItemExtraIntProperty[Ex]Ed[572];
- SetCurrentItemExtra**Str**Property[567];

See also methods of TCustomRichView:

- SetItemExtraIntProperty[Ex][366].

6.1.2.2.115 TRichViewEdit.SetCurrentItemExtraStrProperty -Ex

The methods change a value of a string property for the item at the position of caret

```
procedure SetCurrentItemExtraStrProperty(
  Prop: TRVExtraItemStrProperty[1072];
  const Value: TRVUnicodeString[1099]; AutoReformat: Boolean);
procedure SetCurrentItemExtraStrPropertyEx(Prop: Integer;
  const Value: TRVUnicodeString[1099]; AutoReformat: Boolean);
```

(introduced in versions 1.9 and 15; changed in version 18)

SetCurrentItemExtraStrProperty[Ex](...) is equivalent to TopLevelEditor[489].**SetItemExtraStrProperty[Ex]Ed**[573](TopLevelEditor[489].CurItemNo[480], ...).

This method sets a new **Value** of the item's property identified by **Prop**.

In **SetCurrentItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[1072]. 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[1124]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[582].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[508] 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[503] and EndItemModify[511]);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call Reformat[339]).

Method type: editing-style.

See also methods:

- GetCurrentItemExtraStrProperty[Ex][519];

- SetItemExtraStrProperty[Ex]Ed[573];
- SetCurrentItemExtra**Int**Property[Ex][566].

See also methods of TCustomRichView:

- SetItemExtraStrProperty[Ex][367].

6.1.2.2.116 TRichViewEdit.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[1056]);
procedure SetCurrentItemTextW(const s: TRVUnicodeString[1099]);
```

(Introduced in version 1.4, 1.7)

- **SetCurrentItemText**(s) is equivalent to TopLevelEditor[489].**SetItemTextEd**[574](TopLevelEditor[489].CurItemNo[480], s).
- **SetCurrentItemTextA**(s) is equivalent to TopLevelEditor[489].**SetItemTextEdA**[574](TopLevelEditor[489].CurItemNo[480], s).
- **SetCurrentItemTextW**(s) is equivalent to TopLevelEditor[489].**SetItemTextEdW**[574](TopLevelEditor[489].CurItemNo[480], 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** converts ANSI to Unicode. If the current item is a text item, a code page for conversion is calculated basing on the its Charset[809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[260].DefCodePage[745] is used.

SetCurrentItemText works:

- like **SetCurrentItemTextA**, in Delphi 2007 and older
- like **SetCurrentItemTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus[160]

Methods type: editing-style.

See also methods:

- GetCurrentItemText -A -W[519];
- SetItemTextEd -A -W[574].

See also methods of TCustomRichView:

- SetItemText -A -W[368].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Obtaining RichView items [116];
- Item types [90];
- "Tags" [96];
- Unicode in RichView [135].

6.1.2.2.117 TRichViewEdit.SetCurrentItemVAlign

Changes vertical alignment (position relative to the line) of the item at the position of caret.

```
procedure SetCurrentItemVAlign(VAlign: TRVVAlign[1100]);
```

(introduced in version 12)

SetCurrentItemVAlign(VAlign) is equivalent to TopLevelEditor[489].**SetItemVAlignEd**[575] (TopLevelEditor[489].CurItemNo[480], VAlign).

Method type: editing-style.

This method must not be called if the current item is one of: text items[167], tables[179], *breaks*[173], tabs[168], list markers[180], footnotes[185] and endnotes[184].

See also methods:

- GetCurrentItemVAlign[520];
- SetItemVAlignEd[575].

See also properties:

- CurItemNo[480].

See also:

- Modifying RichView items[117].

6.1.2.2.118 TRichViewEdit.SetCurrentPictureInfo

Modifies the item at the position of caret, if this item is a picture[169] or *hot-picture*[172].

```
procedure SetCurrentPictureInfo(const AName: TRVUnicodeString[1099];
  Agr: TRVGraphic[1032]; AVAlign: TRVVAlign[1100]; const ATag: TRVTag[1096]);
```

(changed in version 18)

SetCurrentPictureInfo(...) is equivalent to TopLevelEditor[489].**SetPictureInfoEd**[576] (TopLevelEditor[489].CurItemNo[480], ...).

This method can be used if CurItemStyle[481] returns *rvsPicture*[1127].

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW[568] 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[522] (the image is shared if *rvepShared* extra item property[1067] is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetCurrentPictureInfo[522], 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[522] or GetCurrentTag[522] for this item. The *tag* can also be set by SetCurrentTag[570] method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[566] and SetCurrentItemExtraStrProperty[567].

See also methods:

- GetCurrentPictureInfo[522];
- SetPictureInfoEd[576];
- SetCurrentItemExtraIntProperty[566].

See also properties:

- CurItemNo[480];
- CurItemStyle[481].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.119 TRichViewEdit.SetCurrentTag

Changes tag of the item at the position of caret.

```
procedure SetCurrentTag(ATag: TRVTag[1096]);
```

SetCurrentTag(Tag) is equivalent to TopLevelEditor[489].**SetItemTagEd**[574](TopLevelEditor[489].CurItemNo[480], Tag).

Method type: editing-style.

See also methods:

- GetCurrentTag[522];
- SetItemTagEd[574].

See also properties:

- CurItemNo[480].

See also:

- Modifying RichView items[117];
- "Tags"[96].

6.1.2.2.120 TRichViewEdit.SetHotspotInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a *hotspot*[178].

VCL and LCL:

```
procedure SetHotspotInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString[1099];
```

```
  AImageIndex,  AHotImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag[1096]);
```

(changed in version 18)

FireMonkey:

```
procedure SetHotspotInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString[1099];
  AImageIndex,  AHotImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag[1096];
  ImageWidth, ImageHeight: TRVStyleLength[1094]);
```

Parameters:

ItemNo – index of the item. The item must be of *hotspot*[178] type (*rvsHotspot*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item.

Items of subdocuments (table cells[951]) 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[372]), and call SetHotspotInfoEd of this inplace-editor. Alternatively, you can use SetCurrentHotspotInfo[565].

AName – name of *hotspot*[178], any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW[574] method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyExEd[572] 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[572] method.

ATag – *tag* of the item. You can use value returned by GetHotspotInfo[303] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTagEd[574] method.

Additional parameters for FireMonkey version:

ImageWidth, ImageHeight – the desired size of the image. The component chooses the image that fits the specified size. The image is not scaled.

Method type: editing-style (unlike SetHotspotInfo[364]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[572] and SetItemExtraStrPropertyEd[573].

When possible, use SetCurrentHotspotInfo[565] instead of this method.

See also methods of TCustomRichView:

- GetHotspotInfo[303];
- GetItemStyle[310];
- SetHotspotInfo[364].

See also methods:

- SetCurrentHotspotInfo[565].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- Item types[90];
- "Tags"[96].

6.1.2.2.121 TRichViewEdit.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[1067]; 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[244]-1. Items of subdocuments (table cells[951]) 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[372]), and call SetItemExtraIntProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraIntProperty[Ex][566].

In **SetItemExtraIntPropertyEd**, **Prop**'s type is TRVExtraItemProperty[1067]. 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[1119]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[582].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[508] 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[503] and EndItemModify[511]);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call Reformat[339]).

Methods type: editing-style (unlike SetItemExtraIntProperty[Ex][366]).

When possible, use SetCurrentItemExtraIntProperty[Ex][566] instead of these methods.

See also methods:

- SetItemExtra**Str**PropertyEd[573];

- SetCurrentItemExtraIntProperty[Ex] [566].

See also methods of TCustomRichView:

- SetItemExtraIntProperty[Ex] [366].

6.1.2.2.122 TRichViewEdit.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[1072]; const Value: TRVUnicodeString[1099];
  AutoReformat: Boolean);
procedure SetItemExtraStrPropertyExEd(ItemNo, Prop: Integer;
  const Value: TRVUnicodeString[1099]; 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[1072]. 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 [1124]

ItemNo – index of the item, from 0 to ItemCount [244]-1.

Items of subdocuments (table cells [951]) 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 [372]), and call SetItemExtraStrProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraStrProperty [567].

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange [582].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change [508] 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 [503] and EndItemModify [511]);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call Reformat [339]).

Method type: editing-style (unlike SetItemExtraStrProperty[Ex] [367]).

When possible, use SetCurrentItemExtraStrProperty[Ex] [567] instead of this method.

See also methods:

- SetCurrentItemExtraStrProperty[Ex] [567];
- SetItemExtra**Int**Property[Ex]Ed [572].

See also methods of TCustomRichView:

- SetItemExtraStrProperty[Ex] [367].

6.1.2.2.123 TRichViewEdit.SetItemTagEd

Changes tag of the **ItemNo**-th item.

```
procedure SetItemTagEd(ItemNo: Integer; const ATag: TRVTag[1096]);
```

Items are indexed from 0 to ItemCount[244]-1.

Items of subdocuments (table cells[951]) 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[372]), and call SetItemTagEd of this inplace-editor. Alternatively, you can use SetCurrentTag[570].

Tag can be an integer value or a pointer to dynamically allocated string, see *Tags*[96].

Method type: editing-style, unlike SetItemTag[367].

When possible, use SetCurrentTag[570] instead of this method.

See also methods of TCustomRichView:

- GetItemTag[310];
- SetItemTag[367].

See also methods:

- SetCurrentTag[570].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- "Tags"[96].

6.1.2.2.124 TRichViewEdit.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[1056]);
procedure SetItemTextEdW(ItemNo: Integer; const s: TRVUnicodeString[1099]);
```

(Introduced in version 1.4, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item.

Items of subdocuments (table cells[951]) 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[372]), and call SetItemTextEd* of this inplace-editor. Alternatively, you can use SetCurrentItemText*[568].

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** 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[809]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[260].DefCodePage[745] is used.

SetItemTextEd works:

- like **SetItemTextEdA**, in Delphi 2007 and older
- like **SetItemTextEdW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus[160]

Methods type: editing-style.

See also methods:

- SetCurrentItemText -A -W[568].

See also methods of TCustomRichView:

- GetItemText -A -W[310];
- SetItemText -A -W[368].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117];
- "Tags"[96];
- Unicode in RichView[135].

6.1.2.2.125 TRichViewEdit.SetItemVAlignEd

Changes vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
procedure SetItemVAlignEd(ItemNo: Integer; VAlign: TRVVAlign[1100]);
```

(introduced in version 12)

Items are indexed from 0 to ItemCount[244]-1.

Items of subdocuments (table cells[951]) 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[372]), and call SetItemVAlignEd of this inplace-editor. Alternatively, you can use SetCurrentItemVAlign[569].

Method type: editing-style, unlike SetItemVAlign[369].

This method must not be called for text items[167], tables[179], *breaks*[173], tabs[168], list markers[180], footnotes[185] and endnotes[184].

When possible, use SetCurrentItemVAlign[569] instead of this method.

See also methods of TCustomRichView:

- GetItemVAlign[311];
- SetItemVAlign[369].

See also methods:

- SetCurrentItemVAlign[569].

See also properties:

- ItemCount[244].

See also:

- Modifying RichView items[117].

6.1.2.2.126 TRichViewEdit.SetPictureInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a picture[169] or *hot-picture*[172].

```
procedure SetPictureInfoEd(ItemNo: Integer; const AName: TRVUnicodeString[1099];
  Agr: TRVGraphic[1032]; AVAlign: TRVVAlign[1100]; const ATag: TRVTag[1096]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture[169] or *hot-picture*[172] type (*rvsPicture or rvsHotPicture*[1127]), otherwise the method raises ERichViewError[1014] exception. Items are indexed from 0 to ItemCount[244]-1, GetItemStyle[310] returns type of item.

Items of subdocuments (table cells[951]) 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[372]), and call SetPictureInfoEd of this inplace-editor. Alternatively, you can use SetCurrentPictureInfo[569].

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemTextEd[574] 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[318] for this item (the image is shared if *rvepShared* extra item property[1067] is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetPictureInfo[318] 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[318] or GetItemTag[310] for this item. The *tag* can also be set by SetItemTagEd[574] method.

Method type: editing-style (unlike SetPictureInfo[370]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[572] and SetItemExtraStrPropertyEd[573].

When possible, use SetCurrentPictureInfo[569] instead of this method.

See also methods of TCustomRichView:

- GetPictureInfo[318];
- GetItemStyle[310];
- SetPictureInfo[370].

See also methods:

- SetCurrentPictureInfo [569].

See also properties:

- ItemCount [244].

See also:

- Modifying RichView items [117];
- Item types [90];
- "Tags" [96].

6.1.2.2.127 TRichViewEdit.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 [120].

See also methods:

- BeginUndoGroup [505];
- BeginUndoCustomGroup [504].

6.1.2.2.128 TRichViewEdit.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 [577] and UndoName [578].

This method must be called only when the document is formatted.

This method is called when the user presses **Ctrl**+**Z** (**Command**+**Z** on macOS) or **Alt**+**Backspace**.

Method type: editing-style.

See also:

- Redo [549];
- Undo in RichViewEdit [120].

6.1.2.2.129 TRichViewEdit.UndoAction

Returns type of operation to be undone [577] next.

```
function UndoAction: TRVUndoType [1098];
```

(Introduced in version 1.3)

See TRVUndoType[1098] for the list of operations. If the returned value is *rvutCustom*, you can use UndoName[578].

See also:

- RedoAction[549];
- Undo in RichViewEdit[120].

6.1.2.2.130 TRichViewEdit.UndoName

Returns name of operation to be undone[577] next.

```
function UndoName: TRVUnicodeString[1099];
```

(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[577]).

Returns non-empty string only for custom undo operation initiated by BeginUndoCustomGroup[504].

See also:

- RedoName[550];
- Undo in RichViewEdit[120].

6.1.2.3 Events

Derived from TCustomRichViewEdit

- OnAfterOleDrop[580] [VCL,LCL]
- OnBeforeOleDrop[580] [VCL,LCL]
- OnCaretGetOut[581]
- OnCaretMove[582]
- OnChange[582]
- OnChanging[582]
- OnCheckStickingItems[583]
- OnCurParaStyleChanged[583]
- OnCurTextStyleChanged[584]
- OnDrawCurrentLine[584]
- OnDrawCustomCaret[585] [VCL,LCL]
- OnDropFile[586]
- OnDropFiles[586]
- OnItemTextEdit[588]
- OnMeasureCustomCaret[588]
- OnOleDragEnter[589] [VCL,LCL]
- OnOleDragLeave[589] [VCL,LCL]
- OnOleDragOver[590] [VCL,LCL]
- OnOleDrop[591] [VCL,LCL]
- OnParaStyleConversion[592]
- OnPaste[594]
- OnSaveCustomFormat[595]

- OnSmartPopupClick 595
- OnStyleConversion 595

Derived from TCustomRichView 216

- OnAddStyle 377
- OnAfterDrawImage 378
- OnAssignImageFileName 379
- OnCheckpointVisible 379 (doesn't work in editor)
- OnCMHintShow 380 [VCL,LCL]
- OnControlAction 380
- OnCopy 381
- OnGetItemCursor 382
- OnGetSpellingSuggestions 382 [FMX]
- OnHTMLSaveImage 383
- OnImportFile 385
- OnImportPicture 386
- OnItemAction 387
- OnItemHint 388
- OnJump 390
- OnLoadCustomFormat 390
- OnLoadDocument 391
- OnNewDocument 391
- OnPaint 392
- OnProgress 392
- OnReadField 393
- OnReadHyperlink 395
- OnReadMergeField 398
- OnRVDblClick 399
- OnRVFControlNeeded 400
- OnRVFImageListNeeded 400
- OnRVFPictureNeeded 401
- OnRVMouseDown 402
- OnRVMouseMove 403
- OnRVMouseUp 403
- OnRVRightClick 404
- OnSaveComponentToFile 405
- OnSaveDocXExtra 407
- OnSaveHTMLExtra 409
- OnSaveImage2 410
- OnSaveItemToFile 411
- OnSaveParaToHTML 413
- OnSaveRTFExtra 414
- OnSelect 416
- OnSpellingCheck 416
- OnStyleTemplatesChange 417
- OnTextFound 418
- OnWriteHyperlink 418
- OnWriteObjectProperties 420

Derived from TRVScroller [869]

- OnHScrolled [877]
- OnVScrolled [877]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- OnApplyStyleLookup [FMX]
- OnClick
- OnContextPopup [VCL, LCL]
- OnDragDrop
- OnDragEnd [FMX]
- OnDragEnter [FMX]
- OnDragLeave [FMX]
- OnDragOver
- OnEndDrag [VCL, LCL]
- OnEnter
- OnExit
- OnGesture (D2010+) [VCL, FMX]
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseEnter [FMX]
- OnMouseLeave [FMX]
- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown [VCL, LCL]
- OnMouseWheelUp [VCL, LCL]
- OnResize
- OnResized [FMX]
- OnStartDrag [VCL,LCL]

6.1.2.3.1 TRichViewEdit.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 [591] 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 and CSS files referred in inserted data (RTF or URL).

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

See also:

- Drag&drop in TRichView' [123];

6.1.2.3.2 TRichViewEdit.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	The user pressed **Left** (or **Right**, if BiDiMode [230] =*rvbdRightToLeft*) while the caret is at the beginning of the document
rvdUp	The user pressed **Up** while the caret is in the first line of the document
rvdRight	The user pressed **Right** (or **Left**, if BiDiMode [230] =*rvbdRightToLeft*) while the caret is at the end of the document
rvdDown	The user pressed **Down** while the caret is in the last line of the document.
rvdTop	The user pressed **Ctrl**+**Home**

Direction	The event happens if...
rvdBottom	The user pressed Ctrl*+End

* on MacOS, Command is used instead of Ctrl.

This event can be used to move focus to another control.

6.1.2.3.3 TRichViewEdit.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[356] here). If you need to update user interface reflecting selection, use OnSelect[416].

In this event, you can use the following:

- CurItemNo[480] property;
- OffsetInCurItem[488] property;
- GetCurrentLineCol[521] method.

6.1.2.3.4 TRichViewEdit.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[508].

Viewer-style methods do not generate this event.

See also property:

- Modified[487].

6.1.2.3.5 TRichViewEdit.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[582] event;
- ReadOnly[488] property.

6.1.2.3.6 TRichViewEdit.OnCheckStickingItems

Allows to "stuck" items together.

```
type
  TRVStickingItemsEvent = procedure (Sender: TCustomRichViewEdit;
    RVData: TCustomRVData[1014]; 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[815] property of a text style).

6.1.2.3.7 TRichViewEdit.OnCurParaStyleChanged

Occurs when the index of current paragraph style (value of CurParaStyleNo[481] 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[481].

See also events:

- OnCurTextStyleChanged[584].

6.1.2.3.8 TRichViewEdit.OnCurTextStyleChanged

Occurs when the index of current text style (value of CurTextStyleNo[482] 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[482].

See also events:

- OnCurParaStyleChanged[583].

6.1.2.3.9 TRichViewEdit.OnDrawCurrentLine

Allows to highlight the current line (i.e. the line containing the caret).

VCL and LCL:

```
  TRVDrawCurrentLineEvent = procedure(Sender: TCustomRichViewEdit[469];
    ACanvas: TCanvas; ARVData: TCustomRVFormattedData[1014];
    AStartItemNo, AStartOffs, AEndItemNo, AEndOffs: Integer;
    const ARect: TRVCoordRect[1064]; APrepaint: Boolean) of object;
property OnDrawCurrentLine: TRVDrawCurrentLineEvent;
```

FMX:

```
  TRVDrawCurrentLineEvent = procedure(Sender: TCustomRichViewEdit[469];
    ACanvas: TRVFMXCanvas[1018]; ARVData: TCustomRVFormattedData[1014];
    AStartItemNo, AStartOffs, AEndItemNo, AEndOffs: Integer;
    const ARect: TRVCoordRect[1064]; APrepaint: Boolean) of object;

property OnDrawCurrentLine: TRVDrawCurrentLineEvent;
```

(introduced in version 21)

Parameters:

Canvas – canvas to paint.

ARVData – document containing the current line (**Sender**.RVData[254])

AStartItemNo – index of the first item on this line (in the range from 0 to **ARVData**.ItemCount[244] - 1)

AStartOffs – position of the line start in **AStartItemNo** (0 for a non-text item; 1-based index of the first character on the line for a text item)

AEndItemNo – index of the last item on this line (in the range from 0 to **ARVData**.ItemCount[244] - 1)

AEndOffs – position of the line end in **AEndItemNo** (1 for a non-text item; 1-based index of the last character on the line + 1 for a text item)

ARect – rectangle covering this line

APrepaint – *False*, reserved for future use. This event is always called when all other content is already drawn.

Example: highlighting with semitransparent yellow color

```
procedure TForm1.RichViewEdit1DrawCurrentLine(
  Sender: TCustomRichViewEdit; ACanvas: TRVFMXCanvas;
  ARVData: TCustomRVFormattedData; AStartItemNo,
  AStartOffs, AEndItemNo, AEndOffs: Integer;
  const ARect: TRectF; APrepaint: Boolean);
begin
  if not Prepaint then
    Sender.Style.GraphicInterface.FillRectAlpha(ACanvas, ARect, rvclYellow, 50)
end;
```

6.1.2.3.10 TRichViewEdit.OnDrawCustomCaret

Allows to draw a custom caret.

```
type
  TRVDrawCustomCaretEvent = procedure (Sender: TCustomRichViewEdit[469];
    Canvas: TCanvas; const Rect: TRVCoordRect[1064]) of object;
property OnDrawCustomCaret: TRVDrawCustomCaretEvent;
```

(introduced in version 10)

Custom caret is drawn if CustomCaretInterval[483]>0.

OnMeasureCustomCaret[588] event occurs before each call of this event.

Example

This example draws a rectangle/ellipse caret. In combination with the example in OnMeasureCustomCaret[588] topic, it draws rectangle/ellipse around the character to the right of the caret.

```
procedure TMyForm.MyRichViewEditDrawCustomCaret(
  Sender: TCustomRichViewEdit;
  Canvas: TCanvas; const Rect: TRVCoordRect[1064]);
const f: Boolean=True;
var r: TRVCoordRect[1064];
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;
```

6.1.2.3.11 TRichViewEdit.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[1099]; 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[544]), or as a result of drag&drop[123].

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.

6.1.2.3.12 TRichViewEdit.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[478] 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[586] 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** (how to insert files of all formats supported by Microsoft Office text import converters)[603].

See also:

- Drag&Drop[123].

6.1.2.3.13 TRichViewEdit.OnItemResize

Occurs after resizing image[169], control[174] or table[179] with the mouse.

```
type
  TRVItemResizeEvent = procedure(Sender: TCustomRichViewEdit;
    RVData: TCustomRVFormattedData[1014]; 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[254], table cell[951], or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**.

(**Val1**, **Val2**) are:

- for tables[179]: (0, row index) after resizing row, or (1, column index) after resizing columns;
- for controls[174]: (old control width, old control height);
- for other resizable items (pictures[169], shapes in Report Workshop): (old value of *rvepImageWidth*[1067] property, old value of *rvepImageHeight*[1067] property).

6.1.2.3.14 TRichViewEdit.OnItemTextEdit

Occurs when text item [167]'s text is changed as a result of editing operation.

```
type
  TRVItemTextEditEvent = procedure (Sender: TCustomRichViewEdit;
    const OldText: TRVUnicodeString[1099]; RVData: TCustomRVData[1014];
    ItemNo: Integer; var NewTag: TRVTag[1096]; 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 [254], table cell [951], or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**. You can get text after the editing operation using **RVData**.GetItemTextW [310] (**ItemNo**).

NewTag – the item *tag* [96], the same as **RVData**.GetItemTag [310](**ItemNo**)

NewStyleNo – the item style, index in the collection Style [260].TextStyles [764], the same as **RVData**.GetItemStyle [310](**ItemNo**)

Output Parameters:

NewTag – value of *tag* [96] to assign to this item.

NewStyleNo – style index to assign to this item.

6.1.2.3.15 TRichViewEdit.OnMeasureCustomCaret

Allows to define size and position for a custom caret.

```
type
  TRVMeasureCustomCaretEvent = procedure (Sender: TCustomRichViewEdit;
    var Rect: TRVCoordRect[1064]) of object;
property OnMeasureCustomCaret: TRVMeasureCustomCaretEvent;
```

(introduced in version 10)

Custom caret [VCL and LCL]

A custom caret is drawn if CustomCaretInterval [483]>0.

This event is called before each call to OnDrawCustomCaret [585]. 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 [585]
.

Standard caret

If the global variable RichViewEditCustomCaretSize[1108] = *True*, this even is called for the standard caret as well. If RichViewEditCaretPosition[1108] <> *rvhcpRight*, the editor adjusts the position of the caret after calling this event.

6.1.2.3.16 TRichViewEdit.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[216];
    const DataObject: IDataObject; Shift: TShiftState;
    X, Y: Integer; PossibleDropEffects: TRVOleDropEffects[1081];
    var DropEffect: TRVOleDropEffect[1081]) 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[590] will be called while the user moves the mouse pointer inside the editor. Dragging will be finished either by calling OnOleDrop[591] (successfully) or by calling OnOleDragLeave[589] (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[590];
- OnOleDragLeave[589];
- OnOleDrop[591].

See also:

- Drag&drop in TRichView[123].

6.1.2.3.17 TRichViewEdit.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[589], it occurs when the user moves the mouse pointer into the editor while dragging. After that, OnOleDragOver[590] is called while the user moves the mouse inside the editor. Dragging is finished either by calling OnOleDrop[591] (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[589];
- OnOleDragOver[590];
- OnOleDrop[591].

See also:

- Drag&drop in TRichView[123].

6.1.2.3.18 TRichViewEdit.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[589] (or automatically by the editor).

```
type
  TRVOleDragOverEvent = procedure (Sender: TCustomRichView[216];
    Shift: TShiftState; X, Y: Integer;
    PossibleDropEffects: TRVOleDropEffects[1081];
    var DropEffect: TRVOleDropEffect[1081]) 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[589] occurs before this event, with DataObject in parameters. Dragging will be finished either by calling OnOleDrop[591] (successfully) or by calling OnOleDragLeave[589] (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[589];
- OnOleDragLeave[589];
- OnOleDrop[591].

See also:

- Drag&drop in TRichView[123].

6.1.2.3.19 TRichViewEdit.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[589], it occurs when the user moves the mouse pointer into the editor while dragging. After that, OnOleDragOver[590] is called while the user moves the mouse inside the editor. Dragging is finished either by calling this event (successfully) or by calling OnOleDragLeave[589] (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[580] events.

See also events:

- OnOleDragEnter[589];
- OnOleDragOver[590];
- OnOleDragLeave[589];
- OnDropFiles[586].

See also:

- Drag&drop in TRichView'[123];
- How to make one line, plain text editor[1134].

6.1.2.3.20 TRichViewEdit.OnParaStyleConversion

Occurs while executing ApplyParaStyleConversion[499] 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[260].ParaStyles[757]);

UserData – value passed as a parameter in ApplyParaStyleConversion[499];

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[828];
begin
  // creating paragraph style with the desired attributes
  ParaInfo := TParaInfo.Create(nil);
  try
    ParaInfo.Assign(RVStyle1.ParaStyles[757][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[771](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[499] as UserData;
- RVStyle1 – TRVStyle[739] 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 [804] property is set to *False*. It is important, otherwise DeleteUnusedStyles [293] will not be able to delete this style, even if all paragraphs of this style will be removed.

See demos:
- Demos*\Editors\Editor 2\

See also:
- OnStyleConversion [595].

See also methods:
- ApplyParaStyleConversion [499].

6.1.2.3.21 TRichViewEdit.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 [543] method, or Ctrl+V (Command+V on macOS) / 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 [543] 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 [540](s,False);
  DoDefault := False;
end;
```

See also:
- RichView and Clipboard [113];
- How to make a plain text editor [1134].

6.1.2.3.22 TRichViewEdit.OnSaveCustomFormat

This event allows saving data to field in your own format.

```
type
  TRVCustomFormatEvent = procedure (Sender: TCustomRichView[216];
    Stream: TStream; var DoDefault: Boolean) of object;
property OnSaveCustomFormat: TRVCustomFormatEvent;
```

(introduced in version 10)

This event occurs when saving a document from a database field:

- linked using live binding (Delphi XE2+, see Document[238] property)
- by TDBRichViewEdit[616] component

Stream is empty initially. Save the content of **Sender** in Stream and set **DoDefault** to *False*, otherwise the component will save its document according to:

- (for TRichViewEdit linked using LiveBindings) Document[238].FieldFormat[431]
- (for TDBRichViewEdit[616]) FieldFormat[623] property.

See also events:

- OnLoadCustomFormat[390].

6.1.2.3.23 TRichViewEdit.OnSmartPopupClick

Occurs when user clicks "smart popup"[598] 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[601]). If you use menu, it's not necessary to process this event.

6.1.2.3.24 TRichViewEdit.OnStyleConversion

Occurs while executing ApplyStyleConversion[501] 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[260].TextStyles[764]);

UserData – value passed as a parameter in ApplyStyleConversion[501];

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[482]);

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[815])) then
    NewStyleNo := UserData;
end;
```

This example does exactly the same work as ApplyTextStyle[502], i.e. MyRichViewEdit.ApplyStyleConversion[501](StyleNo) will be equivalent to MyRichViewEdit.ApplyTextStyle[502](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[821];
begin
  // creating text style with the desired attributes
  FontInfo := TFontInfo.Create(nil);
  try
    FontInfo.Assign(RVStyle1.TextStyles[764][StyleNo]);
    case UserData of
      TEXT_BOLD:
        if btnBold.Down then
          FontInfo.Style := FontInfo.Style[817]+[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(771)(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(501) as UserData;
- RVStyle1 – TRVStyle(739) component linked with the richview editor;
- ColorDialog1: TColorDialog;
- FontDialog1: TFontDialog;
- btnBold, btnItalic, btnUnderline: TSpeedButton;
- FontName: String;
- FontSize: Integer.

See demos:

- Demos*\Editors\Editor 2\

See also:

- OnParaStyleConversion(592).

See also methods:

- ApplyStyleConversion(501).

6.1.2.4 Classes of properties

TRVSmartPopupProperties [598] is a type of TRichViewEdit [469].SmartPopupProperties [489] property. It defines properties of small buttons that can pop up next to document items.

6.1.2.4.1 TRVSmartPopupProperties

Properties of "smart popup" buttons (analogs of buttons used for Microsoft Office "smart tags"). Class of TCustomRichViewEdit.SmartPopupProperties [489].

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 [90]. 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 [582] event to set properties of "smart popup" and display the button. The button is displayed when you assign True to TCustomRichViewEdit.SmartPopupVisible [489]. 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 [599];
- Color [599], LineColor [601];
- HoverColor [600], HoverLineColor [600];
- ImageList [600], ImageIndex [600].

Hint [600] is displayed when the user moves the mouse pointer over the button.

Position [601] defines the button position.

When the user clicks the button or presses ShortCut [601], Menu [601] or Popup [601] is displayed (if assigned). Instead of a menu/popup, you can process TCustomRichViewEdit.OnSmartPopupClick [595] event to execute a command or display a dialog.

6.1.2.4.1.1 Properties

In TRVSmartPopupProperties

- ButtonType [599]
- Color [599]

- Hint[600]
- HoverColor[600]
- HoverLineColor[600]
- ImageIndex[600]
- ImageList[600]
- LineColor[601]
- Menu[601]
- Popup[601] [FMX]
- Position[601]
- ShortCut[601]

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[601] 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[582] (to create the proper button for the item at the position of caret)

Default value:

rvsptDropDown

Background color of a button.

```
property Color: TRVColor[1063];
```

Default value:

rvclWindow[1105]

See also:

- LineColor[601];
- HoverColor[600];
- HoverLineColor[600].

Contains the text string that can appear when the user moves the mouse over a button.

```
property Hint: TRVUnicodeString[1099];
```

You can change this property in TCustomRichViewEdit.OnCaretMove[582] (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: TRVColor[1063];
```

Default value:

rvclInfoBk[1105]

See also:

- Color[599];
- LineColor[601];
- HoverLineColor[600].

Color of button border and the right side sign (ellipsis or triangle)[599] when a button is under the mouse pointer.

```
property HoverLineColor: TRVColor[1063];
```

Default value:

rvclInfoText[1105]

See also:

- Color[599];
- HoverColor[600];
- HoverColor[600].

Index of a button image in ImageList[600].

```
property ImageIndex: Integer;
```

This image is displayed on a button.

You can change this property in TCustomRichViewEdit.OnCaretMove[582] (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[600]-th image is shown on button.

You can change this property in TCustomRichViewEdit.OnCaretMove[582] (to create the proper button for the item at the position of caret).

Border color of a button.

```
property LineColor: TRVColor[1063];
```

Default value:

rvclHighlight[1105]

See also:

- Color[599];
- HoverColor[600];
- HoverLineColor[600].

A link to a popup menu to display when the user clicks a button or presses ShortCut[601].

```
property Menu: TPopupMenu;
```

Alternatively, you can:

- process TCustomRichViewEdit.OnSmartPopupClick[595] event to execute a command or display a dialog;
- (in FireMonkey) assign Popup[601] property.

You can change this property in TCustomRichViewEdit.OnCaretMove[582] (to create the proper button for the item at the position of the caret).

FireMonkey note: this menu works even on Android; TRichView simulates it using TPopup containing TListBox.

If both Popup[601] and **Menu** are assigned, Popup[601] is displayed.

A link to TPopup component to display when the user clicks a button or presses ShortCut[601].

```
property Popup: TPopup;
```

Alternatively, you can:

- process TCustomRichViewEdit.OnSmartPopupClick[595] event to execute a command or display a dialog;
- assign Menu[601] property.

You can change this property in TCustomRichViewEdit.OnCaretMove[582] (to create the proper button for the item at the position of the caret).

If both **Popup** and Menu[601] are assigned, **Popup** is displayed.

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 [601].

See also events of TCustomRichViewEdit:

- OnSmartPopupClick [595].

6.1.2.4.1.2 Methods

In TRVSmartPopupProperties

SetButtonState [602]

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 [595].

6.1.2.5 Examples

TRichViewEdit examples

- Moving caret to the beginning of paragraph [602]
- Using OnDropFiles to accept files of different formats [603]

6.1.2.5.1 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: TCustomRichViewEdit[469];
  ParagraphIndex: Integer);
var i: Integer;
begin
  for i := 0 to rve.ItemCount[244]-1 do
  begin
    if rve.IsParaStart[325](i) then
      dec(ParagraphIndex);
    if ParagraphIndex<0 then begin
      rve.SetSelectionBounds[372](i, rve.GetOffsBeforeItem[317](i),
        i, rve.GetOffsBeforeItem[317](i));
      rve.Invalidate;
      exit;
    end;
  end;
end;
```

Call:

```
GoToParagraph(RichViewEdit1, 7);
```

ParagraphIndex is zero-based.

See also:

- TCustomRichViewEdit[469].MoveCaret[542]

6.1.2.5.2 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[586] 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[687] 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: TRVGraphic[1032];
    Ext: String;
    i: Integer;
  begin
    Result := False;
    try
      // 1. Trying to insert as a graphic
      gr := RVGraphicHandler[1125].LoadFromFile(FileName);
      if gr <> nil then
      begin
        Sender.InsertPicture[534]('', gr, rvvaBaseline[1100]);
        Result := True;
        exit;
      end;
      // 2. Trying to insert as RTF file
      Ext := LowerCase(ExtractFileExt(FileName));
      if Ext='.rtf' then
      begin
        Sender.InsertRTFFromFileEd[536](FileName);
        Result := True;
        exit;
      end;
      // 3. Trying to insert as RVF file
      if Ext='.rvf' then
```

```
      begin
        Sender.InsertRVFFromFileEd(537)(FileName);
        Result := True;
        exit;
      end;
      // 4. Trying to insert as text file
      if Ext='.txt' then
      begin
        Sender.InsertTextFromFile(541)(FileName);
        Result := True;
        exit;
      end;
      // 5. Trying to insert using office converters
      for i := 0 to RVOfficeConverter1.ImportConverters(691).Count-1 do
        if Pos(Ext, RVOfficeConverter1.ImportConverters[i].Filter)>0 then
          if RVOfficeConverter1.ImportRTF(694)(FileName, i) then
          begin
            RVOfficeConverter1.Stream(691).Position := 0;
            Sender.InsertRTFFromFileEd(536)(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;
```

6.1.3 TDBRichView

Use TDBRichView to display hypertext documents stored in a database field.

Unit DBRV;

Syntax

```
TDBRichView = class(TCustomRichView)
```

DBRichView does not display its content if it is not linked to RVStyle component via Style(260) property.

The most important property settings can be done in the component editor(224) for TRichView. In Delphi IDE, right click a TDBRichView object on a form, choose "Settings" in the context menu.

If the application doesn't require the data-aware capabilities of TDBRichView, use TRichView(216) instead, to conserve system resources.

For editing documents in database fields, use TDBRichViewEdit(616).

In Delphi XE2 or newer, there is another solution for data awareness: LiveBindings (see TRichView.Document[238] property).

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[869]
TCustomRichView[216]

Field Type

This component must be linked to BLOB field.

If you use RVF[129] (RichView Format) or DocX, 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 and DocX format in them. However, you can use memo fields to display text, HTML, or RTF (Rich Text Format) fields.

Starting from Delphi 2006, Unicode memo fields (Field[610].DataType=*ftWideMemo*) are supported specially. TDBRichView does not attempt loading RVF and DocX from these fields. It tries to load Unicode RTF (RTF code converted from AnsiString to WideString), HTML or Unicode text from them.

If *rvfoLoadBack* is in RVFOptions[254], the component clears BackgroundPicture[229] before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF[129] (RichView Format),
- RTF (Rich Text Format),
- DocX (Microsoft Word document)
- HTML
- text
- others (using events).

For all field types except for Unicode memos:

1. OnLoadCustomFormat[390] 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 (either normal RTF or RTF stored as WideString).
4. If the field does not contain RTF, it tries to load HTML (including HTML in UTF-16 encoding).
5. If the field does not contain HTML, it tries to load DocX.
6. If the field does not contain DocX:
   a. If AllowMarkdown[608], it loads Markdown

b. Otherwise, it loads a plain text

For Unicode memos:

The sequence is the same, but

- the component does not attempt to load RVF and DocX
- at the final step, it loads Unicode (UTF-16) text or Markdown.

Events

OnNewDocument[391] 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[391] occurs.

OnLoadCustomFormat[390] allows loading documents in custom formats.

6.1.3.1 Properties

In TDBRichView

- AllowMarkdown[608]
- AutoDeleteUnusedStyles[608]
- AutoDisplay[609]
- CodePage[609]
- DataField[610]
- DataSource[610]
- Field[610]

Derived from TCustomRichView

- AnimationMode[228]
- BackgroundPicture[229]
- BackgroundProperties[229]
- BiDiMode[230]
- BottomMargin[231]
- ClearLeft[232]
- ClearRight[232]
- Color[233]
- CPEventKind[233]
- Cursor[234]
- DarkMode[234]
- Delimiters[235]
- DocObjects[235]
- DocParameters[237]
- DocProperties[238]
- DocumentHeight[239]
- DocumentPixelsPerInch[239]
- DoInPaletteMode[241]
- FirstItemVisible[243]
- FirstJumpNo[243]
- HTMLReadProperties[243]

- HTMLSaveProperties 244
- ItemCount 244
- LastItemVisible 244
- LeftMargin 245
- MarkdownProperties 245
- MaxLength 246
- MaxTextWidth 246
- MinTextWidth 246
- NoteText 247
- Options 247
- PageBreaksBeforeItems 250
- RightMargin 251
- RTFOptions 252
- RTFReadProperties 253
- RVData 254
- RVFOptions 254
- RVFParaStylesReadMode 258
- RVFTextStylesReadMode 258
- RVFWarnings 258
- SingleClick 260 (deprecated)
- SelectionHandlesVisible 260 (D2010+)
- **Style** 260
- StyleTemplateInsertMode 260
- TabNavigation 262
- TopMargin 262
- UseStyleTemplates 263
- UseVCLThemes 264
- VAlign 264
- VSmallStep 264
- WordWrap 265

Derived from TRVScroller 869

- BorderStyle 871
- HScrollMax 871
- HScrollPos 872
- HScrollVisible 872
- InplaceEditor 872
- NoVScroll 873
- Tracking 873
- UseXPThemes 873
- VScrollMax 874
- VScrollPos 874
- VScrollVisible 874
- WheelStep 875

Derived from TCustomControl

- Align

- Anchors
- Constraints
- Ctl3D
- DragMode
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- Left
- Name
- ParentCtl3D
- ParentShowHint
- PopupMenu
- ShowHint
- StyleName (D10.4+)
- TabOrder
- TabStop
- Tag
- Top
- Touch (D2010+)
- Visible
- Width

6.1.3.1.1 TDBRichView.AllowMarkdown

Turn on/off Markdown loading.

```
property AllowMarkdown: Boolean;
```

(introduced in version 23)

If the field does not contain RVF, RTF, DocX, or HTML, field content will be loaded:

- as Markdown, if **AllowMarkdown** = *True*
- as a plain text, otherwise

Default value:

False

See also

- CodePage [609]
- TRichView [216].Document [429].AllowMarkdown [430] (for LiveBindings)

6.1.3.1.2 TDBRichView.AutoDeleteUnusedStyles

If this property is set to *True*, the component deletes unused text [764], paragraph [757] and list [755] styles before loading from a DB field.

```
property TDBRichViewEdit.AutoDeleteUnusedStyles: Boolean;
```

(introduced in version 21)

It's highly recommended to assign *True* to this property. However, the following condition are required to auto-delete styles correctly:

- this component must not share TRVStyle[739] component with other TRichView controls (others TRichView controls must not be linked to the same TRVStyle component via Style[260] property) and

Default value:

False

See also

- TRichView[216].Document[238].AutoDeleteUnusedStyles[430] (for live binding)
- TDBRichViewEdit[616].AutoDeleteUnusedStyles[621]

6.1.3.1.3 TDBRichView.AutoDisplay

Determines whether to automatically display the contents of the 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[614] method causes the 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.

6.1.3.1.4 TDBRichView.CodePage

Encoding for loading plain text and Markdown.

```
property CodePage: TRVCodePage[1062];
```

(introduced in version 23)

The default value of this property is *RV_CP_DEFAULT*[1118]. See the topic about this constant for details.

This property is not used when working with WideMemo fields (for these fields, text and Markdown are always oaded as Unicode (UTF-16))

Default value:

RV_CP_DEFAULT[1118]

See also

- TRichView[216].Document[429].CodePage[431] (for LiveBindings)

6.1.3.1.5 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[610] property of the TDBRichView control specifies the dataset which contains the DataField.

Please use only BLOB fields with TDBRichView.

6.1.3.1.6 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[610] property to specify the particular field within the dataset.

6.1.3.1.7 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[129] (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[254]).

6.1.3.2 Methods

In TDBRichView

Create[614]
Destroy[614]
LoadField[614]

Derived from TCustomRichView

AddBreak[269]
AddBullet[270]

AddCheckpoint 271
AddControl 272
AddFmt 273
AddHotPicture 274
AddHotspot 275
AddItem 276
AddNL -A -W 277
AddPicture 278
AddTab 280
AddTextNL -A -W 281
AppendFrom 282
AppendRVFFromStream 283
AssignSoftPageBreaks 284
BeginOleDrag 284
Clear 286
ClearLiveSpellingResults 286
ClearSoftPageBreaks 286
ClientToDocument, DocumentToClient 286
ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits 287
Copy 288
CopyDef 288
CopyImage 289
CopyMarkdown 289
CopyRTF 290
CopyRVF 290
CopyText -A -W 290
Create 291
DeleteItems 291
DeleteMarkedStyles 292
DeleteParas 293
DeleteSection 293
DeleteUnusedStyles 293
Deselect 294
Destroy 294
FindCheckPointByName 294
FindCheckPointByTag 295
FindControlItemNo 295
Format 295
FormatTail 296
GetBreakInfo 296
GetBulletInfo 297
GetCheckpointByNo 298
GetCheckpointInfo 299
GetCheckpointItemNo 299
GetCheckpointNo 299
GetCheckpointXY 300
GetCheckpointY 300

GetCheckpointYEx 301
GetControlInfo 301
GetFirstCheckpoint 302
GetFocusedItem 302
GetHotspotInfo 303
GetItem 304
GetItemAt 305
GetItemCheckpoint 305
GetItemCoords 306
GetItemCoordsEx 307
GetItemExtraIntProperty 307
GetItemExtraIntPropertyEx 307
GetItemExtraStrProperty 308
GetItemExtraStrPropertyEx 308
GetItemNo 309
GetItemPara 309
GetItemStyle 310
GetItemTag 310
GetItemText -A -W 310
GetItemVAlign 311
GetJumpPointLocation 312
GetJumpPointY 313
GetLastCheckpoint 313
GetLineNo 314
GetListMarkerInfo 315
GetNextCheckpoint 315
GetOffsAfterItem 316
GetOffsBeforeItem 317
GetPictureInfo 318
GetPrevCheckpoint 319
GetRealDocumentPixelsPerInch 319
GetSelectedImage 319
GetSelectionBounds 320
GetSelText -W 321
GetTextInfo 321
GetWordAt -A -W 322
InsertRVFFromStream 323
IsFromNewLine 325
IsParaStart 325
LiveSpellingValidateWord 326
LoadDocX 326 (D2009+ or with Delphi ZLib)
LoadDocXFromStream 327 (D2009+ or with Delphi ZLib)
LoadFromFile 328
LoadFromFileEx 328
LoadFromStream 329
LoadFromStreamEx 329
LoadHTML 330

LoadHTMLFromStream 332
LoadMarkdown 332
LoadMarkdownFromStream 333
LoadRTF 334
LoadRTFFromStream 335
LoadRVF 336
LoadRVFFromStream 337
LoadText -W 338
LoadTextFromStream -W 339
MarkStylesInUse 339
RefreshListMarkers 340
Reformat 339
RemoveCheckpoint 340
ResetAnimation 341
SaveDocX 341
SaveDocXToStream 342
SaveHTML 342
SaveHTMLToStream 346
SavePicture 349
SaveRTF 350
SaveRTFToStream 351
SaveRVF 351
SaveRVFToStream 352
SaveText -W 352
SaveTextToStream -W 353
SearchText -A -W 354
SelectAll 355
SelectControl 355
SelectionExists 356
SelectWordAt 356
SetAddParagraphMode 357
SetBreakInfo 358
SetBulletInfo 359
SetCheckpointInfo 360
SetControlInfo 361
SetFooter 362
SetHeader 363
SetHotspotInfo 364
SetItemExtraIntProperty 366
SetItemExtraIntPropertyEx 366
SetItemExtraStrProperty 367
SetItemExtraStrPropertyEx 367
SetItemTag 367
SetItemText -A -W 368
SetItemVAlign 369
SetListMarkerInfo 369

- SetPictureInfo 370
- SetSelectionBounds 372
- StartAnimation 374
- StartLiveSpelling 374
- StopAnimation 375
- StoreSearchResult 375
- UpdatePaletteInfo 375

Derived from TRVScroller 869

- ScrollTo 875

6.1.3.2.1 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 224 (and overridden by default!)

6.1.3.2.2 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 614 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.

6.1.3.2.3 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 609 property is *False*, so that the value of the field is not loaded automatically.

Similar to LoadMemo in TDBRichEdit or TDBMemo.

6.1.3.3 Events

Derived from TCustomRichView

- OnAddStyle 377
- OnAfterDrawImage 378

- OnAssignImageFileName 379
- OnCheckpointVisible 379
- OnCMHintShow 380
- OnControlAction 380
- OnCopy 381
- OnGetItemCursor 382
- OnHTMLSaveImage 383
- OnImportFile 385
- OnImportPicture 386
- OnItemAction 387
- OnItemHint 388
- OnJump 390
- OnLoadCustomFormat 390
- OnLoadDocument 391
- OnNewDocument 391
- OnPaint 392
- OnProgress 392
- OnReadField 393
- OnReadHyperlink 395
- OnReadMergeField 398
- OnRVDblClick 399
- OnRVFControlNeeded 400
- OnRVFImageListNeeded 400
- OnRVFPictureNeeded 401
- OnRVMouseDown 402
- OnRVMouseMove 403
- OnRVMouseUp 403
- OnRVRightClick 404
- OnSaveComponentToFile 405
- OnSaveDocXExtra 407
- OnSaveHTMLExtra 409
- OnSaveImage2 410
- OnSaveItemToFile 411
- OnSaveParaToHTML 413
- OnSaveRTFExtra 414
- OnSelect 416
- OnSpellingCheck 416
- OnStyleTemplatesChange 417
- OnTextFound 418
- OnWriteHyperlink 418
- OnWriteObjectProperties 420

Derived from TRVScroller 869

- OnHScrolled 877
- OnVScrolled 877

Derived from TCustomControl

- OnClick
- OnContextPopup
- OnDragDrop
- OnDragOver
- OnEndDrag
- OnEnter
- OnExit
- OnGesture (D2010+)
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown
- OnMouseWheelUp
- OnResize
- OnStartDrag

6.1.4 TDBRichViewEdit

Use TDBRichView to edit hypertext documents stored in database field.

Unit DBRV;

Syntax

```
TDBRichViewEdit = class(TCustomRichViewEdit 469)
```

DBRichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style 260 property.

The most important property settings can be done in the component editor 224 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 621 to *True*.

If the application doesn't require the data-aware capabilities of TDBRichViewEdit, use TRichViewEdit 469 instead, to conserve system resources.

If the application doesn't require the editing capabilities of TDBRichViewEdit, use TDBRichView 604 instead, to conserve system resources.

In Delphi XE2 or newer, there is another solution for data awareness: LiveBindings (see TRichView.Document 238 property).

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl

TCustomControl
TRVScroller [869]
TCustomRichView [216]
TCustomRichViewEdit [469]

Field Type

This component must be linked to BLOB field.

If you use RVF [129] (RichView Format) or DocX, the database field must have binary BLOB type (i.e. must have ability to store arbitrary binary data, without conversion and filtering of characters).

You can also use text BLOB field type (memo). Memo fields cannot store binary data, so it is impossible to store documents in RVF and DocX format in them. However, you can use memo fields to display text, Markdown, HTML, or RTF (Rich Text Format) fields.

Starting from Delphi 2006, Unicode memo fields (Field [623].DataType=*ftWideMemo*) are supported specially. TDBRichView does not attempt loading RVF and DocX from these fields. It tries to load Unicode RTF (RTF code converted from String to WideString), HTML, Markdown, or Unicode text from them.

If *rvfoLoadBack* is in RVFOptions [254] and FieldFormat [623]=*rvdbRVF*, the component clears BackgroundPicture [229] before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF [129] (RichView Format),
- RTF (Rich Text Format),
- DocX (Microsoft Word document)
- HTML
- Markdown
- text
- others (using events).

For all field types except for Unicode memos:

1. OnLoadCustomFormat [390] 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 (either normal RTF or RTF stored as WideString).
4. If the field does not contain RTF, it tries to load HTML (including HTML in UTF-16 encoding).
5. If the field does not contain HTML, it tries to load DocX.
6. If the field does not contain DocX:
   a. If (FieldFormat [623] = *rvdbMarkdown*) or AllowMarkdown [621], it loads Markdown
   b. Otherwise, it loads a plain text

For Unicode memos:

The sequence is the same, but

- the component does not attempt to load RVF and DocX
- at the final step, it loads Unicode (UTF-16) text or Markdown

Saving data to the field:

OnSaveCustomFormat[595] event allows saving data in your own format. If this event is not processed, documents are saved in FieldFormat[623].

Events

OnNewDocument[391] 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[391] occurs.

OnLoadCustomFormat[390] and OnSaveCustomFormat[595] allow saving and loading documents in custom formats.

Examples

Example 1: Using viewer-style methods in DBRichViewEdit[634] (such as LoadRTF[334])

Example 2: Inserting controls what can modify themselves in DBRichViewEdit[634] (for example, inserting TEdit; user can modify its text)

6.1.4.1 Properties

In TDBRichViewEdit

- AllowMarkdown[621]
- AutoDeleteUnusedStyles[621]
- AutoDisplay[622]
- CodePage[622]
- DataField[622]
- DataSource[623]
- Field[623]
- FieldFormat[623]
- IgnoreEscape[624]
- ReadOnly[624]

Derived from TCustomRichViewEdit[469]

- AcceptDragDropFormats[478]
- AcceptPasteFormats[478]
- ActualCurTextStyleNo[480]
- CurItemNo[480]
- CurItemStyle[481]
- CurParaStyleNo[481]
- CurTextStyleNo[482]
- CustomCaretInterval[483]
- DefaultPictureVAlign[483]
- EditorOptions[483]
- ForceFieldHighlight[486]
- LiveSpellingMode[487]
- Modified[487]
- OffsetInCurItem[488]
- ReadOnly[488]

- SmartPopupProperties 489
- SmartPopupVisible 489
- TopLevelEditor 489
- UndoLimit 490

Derived from TCustomRichView 216

- AnimationMode 228
- BackgroundPicture 229
- BackgroundProperties 229
- BiDiMode 230
- BottomMargin 231
- ClearLeft 232
- ClearRight 232
- Color 233
- CPEventKind 233
- Cursor 234
- DarkMode 234
- Delimiters 235
- DocObjects 235
- DocParameters 237
- DocProperties 238
- DocumentHeight 239
- DocumentPixelsPerInch 239
- DoInPaletteMode 241
- FirstItemVisible 243
- FirstJumpNo 243
- HTMLReadProperties 243
- HTMLSaveProperties 244
- ItemCount 244
- LastItemVisible 244
- LeftMargin 245
- MarkdownProperties 245
- MaxLength 246
- MaxTextWidth 246
- MinTextWidth 246
- NoteText 247
- Options 247
- PageBreaksBeforeItems 250
- RightMargin 251
- RTFOptions 252
- RTFReadProperties 253
- RVData 254
- RVFOptions 254
- RVFParaStylesReadMode 258
- RVFTextStylesReadMode 258
- RVFWarnings 258
- SingleClick 260 (deprecated)
- SelectionHandlesVisible 260 (D2010+)

- **Style** 260
- StyleTemplateInsertMode 260
- TabNavigation 262
- TopMargin 262
- UseStyleTemplates 263
- UseVCLThemes 264
- VAlign 264
- VSmallStep 264
- WordWrap 265

Derived from TRVScroller 869

- BorderStyle 871
- HScrollMax 871
- HScrollPos 872
- HScrollVisible 872
- InplaceEditor 872
- NoVScroll 873
- Tracking 873
- UseXPThemes 873
- VScrollMax 874
- VScrollPos 874
- VScrollVisible 874
- WheelStep 875

Derived from TCustomControl

- Align
- Anchors
- Constraints
- Ctl3D
- DragMode
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- Left
- Name
- ParentCtl3D
- ParentShowHint
- PopupMenu
- ShowHint
- StyleName (D10.4+)
- TabOrder
- TabStop
- Tag
- Top

- Touch (D2010+)
- Visible
- Width

6.1.4.1.1 TDBRichViewEdit.AllowMarkdown

Turn on/off Markdown loading.

```
property AllowMarkdown: Boolean;
```

(introduced in version 23)

If the field does not contain RVF, RTF, DocX, or HTML, field content will be loaded:

- as Markdown, if (**AllowMarkdown** = *True*) or (FieldFormat[623] = *rvdbMarkdown*)
- as a plain text, otherwise

Default value:

False

See also

- CodePage[622]
- TRichViewEdit[469].Document[429].AllowMarkdown[430] (for LiveBindings)

6.1.4.1.2 TDBRichViewEdit.AutoDeleteUnusedStyles

If this property is set to *True*, the editor deletes unused text[764], paragraph[757] and list[755] styles before posting to the database field.

```
property AutoDeleteUnusedStyles: Boolean;
```

(introduced in version 1.7)

If FieldFormat[623] = *rvdbRTF, rvdbDocX, rvdbHTML*, unused styles are auto-deleted even if **AutoDeleteUnusedStyles** = *False*.

It's highly recommended to assign *True* to this property. However, the following condition are required to auto-delete styles correctly:

- this editor must not share TRVStyle[739] component with other TRichView controls (others TRichView controls must not be linked to the same TRVStyle component via Style[260] property) and
- collections of text, paragraph and list styles must be saved together with documents in the RVF field (see RVFOptions[254], *rvfoSaveTextStyles* and *rvfoSaveParaStyles* must be included for AutoDeleteUnusedStyles = *True*).

Default value:

False

See also

- TRichView[216].Document[238].AutoDeleteUnusedStyles[430] (for LiveBindings)
- TDBRichView[604].AutoDeleteUnusedStyles[608]

6.1.4.1.3 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[631] 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.

6.1.4.1.4 TDBRichViewEdit.CodePage

Encoding for saving and loading plain text and Markdown.

```
property CodePage: TRVCodePage[1062];
```

(introduced in version 23)

The default value of this property is *RV_CP_DEFAULT*[1118]. See the topic about this constant for details.

This property is not used when working with WideMemo fields (for these fields, text and Markdown are always saved/loaded as Unicode (UTF-16))

Default value:

RV_CP_DEFAULT[1118]

See also

- FieldFormat[623]
- TRichViewEdit[469].Document[429].CodePage[431] (for LiveBindings)

6.1.4.1.5 TDBRichViewEdit.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[623] property of the TDBRichView control specifies the dataset which contains the DataField.

Please use only BLOB fields with TDBRichViewEdit.

6.1.4.1.6 TDBRichViewEdit.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[622] property to specify the particular field within the dataset.

6.1.4.1.7 TDBRichViewEdit.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[129] (RichView Format) or DocX, 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 field to display text, Markdown or RTF (Rich Text Format) documents.

See also

- FieldFormat[623]

6.1.4.1.8 TDBRichViewEdit.FieldFormat

Format for saving data in the database field.

```
property FieldFormat: TRVDBFieldFormat[1064];
```

(introduced in version 1.6)

This property allows changing the field saving format to RVF (RichView Format)[129], RTF, DocX, HTML, Markdown, or plain text.

Note: It is not guaranteed that document saved in RTF, DocX, HTML, Markdown and text will be the exactly same after reloading (some item types, text and paragraph attributes cannot be saved in RTF, DocX, HTML or plain text). To store all attributes of text and objects, use RVF format.

You can save data in your own format using OnSaveCustomFormat[595] event.

The value *rvdbMarkdown* affects loading from field as well: if other formats are not detected, field content is loaded as Markdown instead of text (see also AllowMarkdown[621]).

Default value:

rvdbRVF

See also

- TRichViewEdit[469].Document[238].FieldFormat[431] (for LiveBindings)

6.1.4.1.9 TDBRichViewEdit.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

See also:

- TRichViewEdit[469].Document[429].IgnoreEscape[432] (for LiveBindings)

6.1.4.1.10 TDBRichViewEdit.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[488] property is overridden in TDBRichViewEdit to get/set read-only mode in the linked dataset.

6.1.4.2 Methods

In TDBRichViewEdit

Change[631]
Create[631]
Destroy[631]
LoadField[631]

Derived from TCustomRichViewEdit

AdjustControlPlacement[497]
AdjustControlPlacement2[497]
ApplyListStyle[498]
ApplyParaStyle[498]
ApplyParaStyleConversion[499]
ApplyParaStyleTemplate[500]
ApplyStyleConversion[501]
ApplyStyleTemplate[501]
ApplyTextStyle[502]
ApplyTextStyleTemplate[502]
BeginItemModify[503]

BeginUndoCustomGroup 504
BeginUndoCustomGroup2 504
BeginUndoGroup 505
BeginUndoGroup2 505
BeginUpdate 506
CanChange 506
CanPaste 507
CanPasteHTML 507
CanPasteRTF 507
CanPasteRVF 507
Change 508
ChangeListLevels 508
ChangeStyleTemplates 508
Clear 509
ClearTextFlow 509
ClearUndo 509
ConvertToHotPicture 510
ConvertToPicture 510
Create 510
CutDef 511
DeleteSelection 511
Destroy 511
EndItemModify 511
EndUndoGroup2 505
EndUndoCustomGroup2 504
EndUpdate 512
GetCheckpointAtCaret 512
GetCurrentBreakInfo 512
GetCurrentBulletInfo 513
GetCurrentCheckpoint 514
GetCurrentControlInfo 515
GetCurrentHotspotInfo 515
GetCurrentItemExtraIntProperty 518
GetCurrentItemExtraIntPropertyEx 518
GetCurrentItemExtraStrProperty 519
GetCurrentItemExtraStrPropertyEx 519
GetCurrentItem 517
GetCurrentItemEx 517
GetCurrentItemText -A -W 519
GetCurrentItemVAlign 520
GetCurrentLineCol 521
GetCurrentMisspelling 521
GetCurrentPictureInfo 522
GetCurrentTag 522
GetCurrentTextInfo 523
InsertBreak 523
InsertBullet 524

InsertCheckpoint 525
InsertControl 526
InsertDocXFromFileEd 526 (D2009+ or with Delphi ZLib)
InsertDocXFromStreamEd 527 (D2009+ or with Delphi ZLib)
InsertHotPicture 528
InsertHotspot 528
InsertHTMLFromFileEd 529
InsertHTMLFromStreamEd 530
InsertItem 531
InsertMarkdownFromFileEd 532
InsertMarkdownFromStreamEd 532
InsertOEMTextFromFile 533
InsertPageBreak 534
InsertPicture 534
InsertRTFFromFileEd 536
InsertRTFFromStreamEd 536
InsertRVFFromFileEd 537
InsertRVFFromStreamEd 538
InsertStringTag -A -W 539
InsertTab 540
InsertText -A -W 540
InsertTextFromFile -W 541
MoveCaret 542
Paste 543
PasteBitmap 544
PasteHTML 545
PasteGraphicFile 544
PasteMarkdown 546
PasteMetafile 546
PasteRTF 547
PasteRVF 547
PasteText -A -W 548
PasteURL 549
Redo 549
RedoAction 549
RedoName 550
RemoveCheckpointAtCaret 550
RemoveCheckpointEd 550
RemoveCurrentCheckpoint 551
RemoveCurrentPageBreak 551
RemoveLists 552
ResizeControl 552
ResizeCurrentControl 552
SearchText -A -W 553
SelectCurrentLine 554

SelectCurrentWord 555
SetBackgroundImageEd 556
SetBackgroundPropertyEd 557
SetBreakInfoEd 557
SetBulletInfoEd 558
SetCheckpointInfoEd 560
SetControlInfoEd 560
SetCurrentBreakInfo 561
SetCurrentBulletInfo 562
SetCurrentCheckpointInfo 563
SetCurrentControlInfo 564
SetFooter 362
SetHeader 363
SetCurrentHotspotInfo 565
SetCurrentItemExtraIntProperty 566
SetCurrentItemExtraIntPropertyEx 566
SetCurrentItemExtraStrProperty 567
SetCurrentItemExtraStrPropertyEx 567
SetCurrentItemText -A -W 568
SetCurrentItemVAlign 569
SetCurrentPictureInfo 569
SetCurrentTag 570
SetFloatPropertyEd 555
SetHotspotInfoEd 570
SetIntPropertyEd 555
SetItemExtraIntPropertyEd 572
SetItemExtraIntPropertyExEd 572
SetItemExtraStrPropertyEd 573
SetItemExtraStrPropertyExEd 573
SetItemTagEd 574
SetItemTextEd -A -W 574
SetItemVAlignEd 575
SetPictureInfoEd 576
SetStrPropertyEd 555
SetUndoGroupMode 577
Undo 577
UndoAction 577
UndoName 578

Derived from TCustomRichView 216

AddBreak 269
AddBullet 270
AddCheckpoint 271
AddControl 272
AddFmt 273
AddHotPicture 274

AddHotspot 275
AddItem 276
AddNL -A -W 277
AddPicture 278
AddTab 280
AddTextNL -A -W 281
AppendFrom 282
AppendRVFFromStream 283
AssignSoftPageBreaks 284
BeginOleDrag 284
Clear 286
ClearLiveSpellingResults 286
ClearSoftPageBreaks 286
ClientToDocument, DocumentToClient 286
ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits 287
Copy 288
CopyDef 288
CopyImage 289
CopyMarkdown 289
CopyRTF 290
CopyRVF 290
CopyText -A -W 290
Create 291
DeleteItems 291
DeleteMarkedStyles 292
DeleteParas 293
DeleteSection 293
DeleteUnusedStyles 293
Deselect 294
Destroy 294
FindCheckPointByName 294
FindCheckPointByTag 295
FindControlItemNo 295
Format 295
FormatTail 296
GetBreakInfo 296
GetBulletInfo 297
GetCheckpointByNo 298
GetCheckpointInfo 299
GetCheckpointItemNo 299
GetCheckpointNo 299
GetCheckpointXY 300
GetCheckpointY 300
GetCheckpointYEx 301
GetControlInfo 301
GetFirstCheckpoint 302
GetFocusedItem 302

GetHotspotInfo 303
GetItem 304
GetItemAt 305
GetItemCheckpoint 305
GetItemCoords 306
GetItemCoordsEx 307
GetItemExtraIntProperty 307
GetItemExtraIntPropertyEx 307
GetItemExtraStrProperty 308
GetItemExtraStrPropertyEx 308
GetItemNo 309
GetItemPara 309
GetItemStyle 310
GetItemTag 310
GetItemText -A -W 310
GetItemVAlign 311
GetJumpPointLocation 312
GetJumpPointY 313
GetLastCheckpoint 313
GetLineNo 314
GetListMarkerInfo 315
GetNextCheckpoint 315
GetOffsAfterItem 316
GetOffsBeforeItem 317
GetPictureInfo 318
GetPrevCheckpoint 319
GetRealDocumentPixelsPerInch 319
GetSelectedImage 319
GetSelectionBounds 320
GetSelText -W 321
GetTextInfo 321
GetWordAt -A -W 322
InsertRVFFromStream 323
IsFromNewLine 325
IsParaStart 325
LiveSpellingValidateWord 326
LoadDocX 326 (D2009+ or with Delphi ZLib)
LoadDocXFromStream 327 (D2009+ or with Delphi ZLib)
LoadFromFile 328
LoadFromFileEx 328
LoadFromStream 329
LoadFromStreamEx 329
LoadHTML 330
LoadHTMLFromStream 332
LoadMarkdown 332
LoadMarkdownFromStream 333
LoadRTF 334

LoadRTFFromStream 335
LoadRVF 336
LoadRVFFromStream 337
LoadText -W 338
LoadTextFromStream -W 339
MarkStylesInUse 339
RefreshListMarkers 340
Reformat 339
RemoveCheckpoint 340
ResetAnimation 341
SaveDocX 341
SaveDocXToStream 342
SaveHTML 342
SaveHTMLToStream 346
SavePicture 349
SaveRTF 350
SaveRTFToStream 351
SaveRVF 351
SaveRVFToStream 352
SaveText -W 352
SaveTextToStream -W 353
SearchText -A -W 354
SelectAll 355
SelectControl 355
SelectionExists 356
SelectWordAt 356
SetAddParagraphMode 357
SetBreakInfo 358
SetBulletInfo 359
SetCheckpointInfo 360
SetControlInfo 361
SetHotspotInfo 364
SetItemExtraIntProperty 366
SetItemExtraIntPropertyEx 366
SetItemExtraStrProperty 367
SetItemExtraStrPropertyEx 367
SetItemTag 367
SetItemText -A -W 368
SetItemVAlign 369
SetListMarkerInfo 369
SetPictureInfo 370
SetSelectionBounds 372
StartAnimation 374
StartLiveSpelling 374
StopAnimation 375
StoreSearchResult 375
UpdatePaletteInfo 375

Derived from TRVScroller [869]

ScrollTo [875]

6.1.4.2.1 TDBRichViewEdit.Change

Generates OnChange [582] event.

```
procedure Change; override;
```

TCustomRichViewEdit.Change [508] 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 [634].

6.1.4.2.2 TDBRichViewEdit.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 [224] (and overridden by default!)

6.1.4.2.3 TDBRichViewEdit.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 [631] 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.

6.1.4.2.4 TDBRichViewEdit.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 [622] property is *False*, so that the value of the field is not loaded automatically.

Similar to LoadMemo in TDBRichEdit or TDBMemo.

6.1.4.3 Events

In TDBRichViewEdit

- OnSaveCustomFormat 595

Derived from TCustomRichViewEdit

- OnAfterOleDrop 580
- OnBeforeOleDrop 580
- OnCaretGetOut 581
- OnCaretMove 582
- OnChange 582
- OnChanging 582
- OnCheckStickingItems 583
- OnCurParaStyleChanged 583
- OnCurTextStyleChanged 584
- OnDrawCurrentLine 584
- OnDrawCustomCaret 585
- OnDropFile 586
- OnDropFiles 586
- OnItemTextEdit 588
- OnMeasureCustomCaret 588
- OnOleDragEnter 589
- OnOleDragLeave 589
- OnOleDragOver 590
- OnOleDrop 591
- OnParaStyleConversion 592
- OnPaste 594
- OnSaveCustomFormat 595
- OnSmartPopupClick 595
- OnStyleConversion 595

Derived from TCustomRichView 216

- OnAddStyle 377
- OnAfterDrawImage 378
- OnAssignImageFileName 379
- OnCheckpointVisible 379 (doesn't work in editor)
- OnCMHintShow 380
- OnControlAction 380
- OnCopy 381
- OnGetItemCursor 382
- OnHTMLSaveImage 383
- OnImportFile 385
- OnImportPicture 386
- OnItemAction 387
- OnItemHint 388

- OnJump 390
- OnLoadCustomFormat 390
- OnLoadDocument 391
- OnNewDocument 391
- OnPaint 392
- OnProgress 392
- OnReadField 393
- OnReadHyperlink 395
- OnReadMergeField 398
- OnRVDblClick 399
- OnRVFControlNeeded 400
- OnRVFImageListNeeded 400
- OnRVFPictureNeeded 401
- OnRVMouseDown 402
- OnRVMouseMove 403
- OnRVMouseUp 403
- OnRVRightClick 404
- OnSaveComponentToFile 405
- OnSaveDocXExtra 407
- OnSaveHTMLExtra 409
- OnSaveImage2 410
- OnSaveItemToFile 411
- OnSaveParaToHTML 413
- OnSaveRTFExtra 414
- OnSelect 416
- OnSpellingCheck 416
- OnStyleTemplatesChange 417
- OnTextFound 418
- OnWriteHyperlink 418
- OnWriteObjectProperties 420

Derived from TRVScroller 869

- OnHScrolled 877
- OnVScrolled 877

Derived from TCustomControl

- OnClick
- OnContextPopup
- OnDragDrop
- OnDragOver
- OnEndDrag
- OnEnter
- OnExit
- OnGesture (D2010+)
- OnKeyDown
- OnKeyPress
- OnKeyUp

- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown
- OnMouseWheelUp
- OnResize
- OnStartDrag

6.1.4.4 Examples

TDBRichViewEdit examples

- Using viewer-style methods in DBRichViewEdit [634]
- Inserting controls that can modify themselves in DBRichViewEdit [634]

6.1.4.4.1 Using viewer-style methods in DBRichViewEdit

This example shows how to use viewer-style methods in DBRichViewEdit (i.e. methods, introduced in TCustomRichView [216]).

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 [506] then
  begin
    DBRichViewEdit1.Clear [509];
    DBRichViewEdit1.LoadRTF [334](OpenDialog1.FileName);
    DBRichViewEdit1.Change [631];
    DBRichViewEdit1.Format [295];
  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 [536](OpenDialog1.FileName);
end;
```

6.1.4.4.2 Inserting controls that can modify themselves in DBRichViewEdit

This example shows how to work with TEdit inserted in TDBRichViewEdit [616], 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[506] then
    DBRichViewEdit1.Change[631]
  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[526]('', Edit, rvvaBaseline);
end;
```

3) You'll need to assign this event to all edits loaded from the database field with RichView documents. Use OnControlAction[380] 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]);
```

6.1.5 TRVPrintPreview

This component displays a print preview for TRichView[216], TRichViewEdit[469], TDBRichView[604], or TDBRichViewEdit[616].

TRVPrintPreview control can display preview only if it is linked to RVPrint[638], and RVPrint is ready (it has a document assigned and pages formatted)

Unit [VCL/FMX] RVPP / fmxRVPP.

Syntax

```
TRVPrintPreview = class(TCustomRVPrintPreview[888])
```

Hierarchy

TObject
TPersistent
TComponent
TControl

TWinControl
TCustomControl
TRVScroller [869]
TCustomRVPrintPreview [888]

Information

See more info in the topic about the ancestor of TRVPrintPreview, TCustomRVPrintPreview [888].

The main property of this component is RVPrint [638], allowing to link it with TRVPrint [696] object.

In the Lazarus for Linux version, it's highly recommended to assign CachePageImage [638] = *True*.

6.1.5.1 Properties

In TRVPrintPreview

- CachePageImage [638] [VCL,LCL]
- RVPrint [638]

Derived from TCustomRVPrintPreview [888]

- ClickMode [891]
- Color [891] [VCL, LCL]
- DarkModeUI [892]
- Fill [891] [FMX]
- MarginsPen [892] [VCL,LCL]
- MarginsStroke [892] [FMX]
- PageBorderColor [893]
- PageBorderWidth [893]
- PageNo [893]
- PrintableAreaPen [893] [VCL,LCL]
- PrintableAreaStroke [893] [FMX]
- ShadowColor [894]
- ShadowWidth [894]
- ZoomInCursor [895]
- ZoomMode [895]
- ZoomOutCursor [896]
- ZoomPercent [896]

Derived from TRVScroller [869]

- BorderStyle [871] [VCL]
- HScrollMax [871]
- HScrollPos [872]
- HScrollVisible [872] [VCL,LCL]
- InplaceEditor [872] (not used in this component)
- NoVScroll [873]
- Tracking [873]
- UseXPThemes [873] [VCL,LCL]
- VScrollMax [874]
- VScrollPos [874]

- VScrollVisible[874] [VCL,LCL]
- WheelStep[875]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- Align
- Anchors
- AutoHide [FMX]
- Constraints
- Ctl3D [VCL]
- DisableMouseWheel [FMX]
- DragMode
- EnableDragHighlight [FMX]
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- Left
- Locked [FMX]
- Margins [FMX]
- Name
- Opacity [FMX]
- Padding [FMX]
- ParentCtl3D [VCL]
- ParentShowHint
- PopupMenu
- Position [FMX]
- RotationAngle [FMX]
- RotationCenter [FMX]
- Scale [FMX]
- ShowHint
- ShowScrollBars [FMX]
- ShowSizeGrip [FMX]
- Size [FMX]
- StyleLookup [FMX]
- TabOrder
- TabStop
- Touch (D2010+) [VCL,FMX]
- TouchTargetExpansion [FMX]
- Tag
- Top
- Visible
- Width

6.1.5.1.1 TRVPrintPreview.RVPrint

This property links this preview control to TRVPrint component.

```
property RVPrint: TRVPrint[696];
```

TRVPrintPreview control can display preview only if it is linked to RVPrint, and RVPrint is ready (it has a document assigned and pages formatted).

6.1.5.1.2 TRVPrintPreview.CachePageImage

Allows using an intermediate image when drawing a page.

```
property CachePageImage: Boolean;
```

If *False,* a page is drawn by RVPrint[638] directly in this component.

If *True,* a page is drawn in an image; this component displays this image.

The VCL version uses a metafile image.

The Lazarus version uses a bitmap image (at 100% page size, drawn scaled). It's highly recommended to turn this mode on in Lazarus for Linux, it provides better preview quality.

The FireMonkey version does not have this property, it always uses an intermediate bitmap.

Default value

False (see the note about Lazarus for Linux above)

6.1.5.2 Methods

Derived from TCustomRVPrintPreview[888]

- Create[897]
- First[897]
- Last[897]
- Next[897]
- Prev[898]
- SetZoom[898]

Derived from TRVScroller[869]

- ScrollTo[875] (not used in this component)

6.1.5.3 Events

Derived from TCustomRVPrintPreview[888]

- OnZoomChanged[899]

Derived from TRVScroller[869]

- OnHScrolled[877]
- OnVScrolled[877]

Derived from TCustomControl

- OnClick

- OnContextPopup
- OnDragDrop
- OnDragOver
- OnEndDrag
- OnEnter
- OnExit
- OnGesture (D2010+)
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnMouseWheel
- OnMouseWheelDown
- OnMouseWheelUp
- OnStartDrag

6.2 Additional Components

TRichView Components [215] | Additional Components

Icon	Class Name	Description
	TRVStyle [739]	Component for defining visual appearance of TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616] controls. It contains some properties that can be used by several TRichView controls.
	TRVPrint [696]	Component for printing documents contained in TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616].
	TRVReportHelper [721]	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter [687]	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL]
	TRVDataSourceLink [685]	Component allowing assigning TDataSource component to controls inserted in TRichView [216], TRichViewEdit [616], TDBRichView [604], TDBRichViewEdit [616], TSRichViewEdit, TDBSRichViewEdit controls. [VCL and LCL]
	TRVSpellChecker [728]	Component for spelling checking
	TRVChat [640]	A component for implementing a chat in TRichView [216].

6.2.1 TRVChat

This component is designed for displaying a chat with the ability to add, edit, and delete messages, as well as reply to them. Messages can include attached images or files. This is a non-visual component that links to a TRichView [216] component and controls its content and rendering.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChat = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

How to Use

To make the chat look visually appealing, the borders of lines and areas should be smoothed. For the FireMonkey version on mobile platforms, it is recommended to use Skia4Delphi. For the VCL and Lazarus versions, it is recommended to enable rendering with GDI+ (for Delphi XE2+, include RVGDIPlusGrIn unit in "uses" of your unit; for Lazarus, include RVGDIPlusGrInLaz).

Assign a value to ChatRichView [642] – the TRichView [216] component into which messages will be added.

Assign the desired values to the properties that affect the chat's appearance, and then call the AddStyles [646] method (this must be done before starting to use the chat).

Fill the chat's Users [644] collection.

Add messages one by one using AddMessage [645] method.

Handle clicks on the chat and assign HighlightedMessage [642].

Using HighlightedMessage [642], implement message editing, replying, quoting, and deletion:

- editing: ChangeMessageText [647];
- replying and quoting: use TRVChatMessageExtraQuote [673] object in the next AddMessage [645];
- deletion: DeleteMessage [647] or MarkMessageAsDeleted [653]

The instructions above cover the most basic actions available in the chat. To explore the capabilities, see the descriptions of properties, methods, and events, as well as the demo project in **Assorted\Chat** folder.

Important information

The chat is intended for on-screen display. You should not attempt to print ChatRichView [642] to a printer.

A special document item type, TRVChatFileItemInfo, is used to display a file attached to a message (StyleNo [310] = *rvsChatFile* [1127]).

Each message is a single paragraph that may include line breaks. The chat automatically detects URLs in the text.

Limitation

The chat drawing does not work properly in Lazarus for Linux, because its canvas does not have necessary methods (you will see outlines instead of filled figures).

6.2.1.1 Properties

In TRVChat

- Callout 641
- ChatMaxWidth 641
- ChatRichView 642
- DateLine 642
- FileIcons 642
- Fonts 642
- HighlightedMessage 642
- LinkColor 643
- Messages 643
- MessageStatusLine 643
- Quote 643
- Texts 643
- UserPicture 644
- Users 644

6.2.1.1.1 TRVChat.Callout

This property contains sub-properties that define the callout bubble sizes, internal and external paddings, corner roundings, arrow types and sizes, and shadow.

```
property Callout: TRVChatCalloutProperties 657;
```

6.2.1.1.2 TRVChat.ChatMaxWidth

The maximum width of the chat content.

```
property ChatMaxWidth: TRVPixel96Length 1085;
```

A value of 0 means no limit: the chat can take up the entire width of the TRichView window.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

0

6.2.1.1.3 TRVChat.ChatRichView

The TRichView component used for displaying the chat.

```
property ChatRichView: TCustomRichView[216];
```

This component manages the content of this TRichView and draws certain objects in it.

Please use components of the TRichView[216] type. While you can assign a TRichViewEdit[469], it is strongly discouraged because the chat content should not be edited directly.

6.2.1.1.4 TRVChat.DateLine

This property contains sub-properties that define the color, margins, padding, and format for displaying dates of message groups.

```
property DateLine: TRVChatDateProperties[660];
```

Additionally, a font for dates is defined in Fonts[642].DateFont.

6.2.1.1.5 TRVChat.FileIcons

This property contains sub-properties that define the attributes of file icons attached to messages — both the icons drawn automatically by the chat and the images that can be provided instead of them.

```
property FileIcons: TRVChatFileIconsProperties[662];
```

6.2.1.1.6 TRVChat.Fonts

This property contains sub-properties that define fonts used in chat.

```
property Fonts: TRVChatFonts[664];
```

6.2.1.1.7 TRVChat.HighlightedMessage

Selected message.

```
property HighlightedMessage: TRVChatMessage[665];
```

This message is visually highlighted in ChatRichView[642].

This property is convenient for performing operations on a single message (for example, editing, deleting, or quoting it).

See also

- OnHighlightedMessageChange[655]

6.2.1.1.8 TRVChat.LinkColor

Color of hyperlinks in chat.

```
property LinkColor: TRVColor[1063];
```

This property is used by AddStyles[646] when adding a text style used for hyperlinks. Other properties are taken from Fonts[642].MessageFont[664].

If you change value of this property when the chat is already displayed, call UpdateTextStyles[654] to apply the new value.

6.2.1.1.9 TRVChat.Messages

The collection of chat messages.

```
property Messages: TRVChatMessageCollection[668];
```

The item's class is TRVChatMessage[665].

You should not add, delete, or modify the properties of this collection directly. All operations with messages must be performed using the chat's methods. The only exception is changing the message state[667] (to new, sent, read, or failed).

This is a public property, it is not stored in a form.

How to add your fields to messages

Create your own class inherited from TRVChatMessage[665] and assign its type to RVChatMessageClass variable. Do it before the first use of TRVChat component (for example, in the unit's initialization section)

6.2.1.1.10 TRVChat.MessageStatusLine

This property contains sub-properties that define the size, position, and content of the message status line. The status line can include the time, an edited marker, and an icon indicating the message status.

```
property MessageStatusLine: TRVChatMessageStatusLine[676];
```

6.2.1.1.11 TRVChat.Quote

This property contains sub-properties that define how quotes are displayed in messages.

```
property Quote: TRVChatQuoteProperties[679];
```

6.2.1.1.12 TRVChat.Texts

This property contains sub-properties that define text strings used in chat

```
property Texts: TRVChatUITexts[680];
```

6.2.1.1.13 TRVChat.UserPicture

This property contains sub-properties that define how user's pictures (avatars) are displayed in the chat.

```
property UserPicture: TRVChatUserPictureProperties[684];
```

6.2.1.1.14 TRVChat.Users

The collection of chat users (authors of messages).

```
property Users: TRVChatUserCollection[683];
```

The item's class is TRVChatUser[681] or inherited from it.

Each chat message is associated with one of the users (the one who wrote it).

When the chat is already populated, you can add new users to the collection and modify all user properties (except for the name, see below).

You cannot delete users who have messages in the chat.

If you change a user's name, it will only affect new messages.

How to add your fields to users

Create your own class inherited from TRVChatUser[681] and assign its type to RVChatUserClass variable. Do it before the first use of TRVChat component (for example, in the unit's initialization section)

6.2.1.2 Methods

In TRVChat

AddMessage[645]
AddStyles[646]
ChangeMessageImage[646]
ChangeMessageText[647]
ClearMessages[647]
CopyToClipboard[647]
DeleteMessage[647]
FindMessages[648]
GetAutoQuote[648]
GetAutoQuoteFromItemNo[648]
GetFirstVisibleMessage[649]
GetImageAtCursor[649]
GetMessage[649]
GetMessageAndImage[649]
GetMessageAtClientCoords[649]
GetMessageAtCursor[649]
GetMessageAtDocumentCoords[649]
GetMessageAtScreenCoords[649]

GetMessageImage 650
GetMessageImageEx 650
GetMessageItemNo 650
GetMessagesAsText 651
GetMessageTextItemNo 650
GetRangeOfSelectedMessages 651
GetSelectedMessage 651
GetSelectedTextForQuote 652
HasSelectedTextForQuote 652
InsertMessage 652
MarkMessageAsDeleted 653
MoveMessage 653
ScrollToMessage 654
UpdateParaStyles 654
UpdateTextStyles 654

6.2.1.2.1 TRVChat.AddMessage

The methods add a new message to the end of the chat. Two methods with the same functionality, differing only in how parameters are passed – directly or via a record.

```
function AddMessage(AUser: TRVChatUser 681;
  const AText: TRVUnicodeString 1099;
  ADateTime: TDateTime = 0;
  Extra: TRVChatMessageExtra 670 = nil;
  AState: TRVChatMessageState 1062 = rvcmsDefault):
  TRVChatMessage 665;

function AddMessage(
  const Data: TRVChatMessageInsertionInfo 675):
  TRVChatMessage 665;
```

The methods add a new message to the end of the chat. They update both the content of ChatRichView 642 and append an item to the end of Messages 643.

The methods expect ChatRichView 642 to be formatted before they are called. They format the added message.

If ChatRichView 642 was scrolled to the very bottom, it will be scrolled to the end of the newly added message; otherwise, the vertical scroll position remains unchanged.

Parameters

AUser – a reference to the user (the author of this message). This is a reference to one of items in Users 644.

AText – message text. If (**Extra** *nil)* or (Extra is TRVChatMessageExtraQuote), **AText** cannot be empty. **AText** can be multi-line. The chat automatically detects URLs in the text (see RVIsURL 1053).

ADateTime – message time. If 0, the current time is used.

Extra – additional data attached to the message: files, an image, or a quote. It may be an object the following classes:

- TRVChatMessageExtraFiles[670] to attach files
- TRVChatMessageExtraImage[672] to attach an image
- TRVChatMessageExtraQuote[673] to add a quotation
- *nil* to attach nothing.

If **Extra**.AutoDestroy[670] = *True*, this method frees **Extra** (if the message was added successfully)

AState – message status.

Return value

A reference to the appended message on success, *nil* otherwise.

See also:

InsertMessage

6.2.1.2.2 TRVChat.AddStyles

Adds the text and paragraph styles that will be used by the chat to ChatRichView[642].Style[260].

```
procedure AddStyles;
```

This method must be called once before starting to work with the chat. It adds new items to the end of ChatRichView[642].Style's[260] TextStyles[764] and ParaStyles[757] collections.

Later, if you change any chat properties that affect the styles, you should call the UpdateTextStyles and/or UpdateParaStyles method.

6.2.1.2.3 TRVChat.ChangeMessageImage

Replaces the image attached to the message and its associated text.

```
function ChangeMessageImage(Msg: TRVChatMessage[665];
  Image: TRVGraphic[1032];
  const ImageText: TRVUnicodeString[1099]): Boolean;
```

The method updates content of ChatRichView[642].

The method expects ChatRichView[642] to be formatted before they are called. It re-formats the changed message.

Parameters

Msg – the message whose image needs to be replaced. **Msg** must be present in Messages[643] and must already have an attached image – you can only replace an existing image; you cannot add an image to a message that did not have one.

Image – new image. On success, it will be owned by the ChatRichView[642] (do not free it yourself).

ImageText – new text associated with this image.

Return value

True on success.

6.2.1.2.4 TRVChat.ChangeMessageText

Replaces the message text.

```
function ChangeMessageText(Msg: TRVChatMessage[665];
  const AText: TRVUnicodeString[1099];
  EditCount: Integer = -1): Boolean;
```

The method updates content of ChatRichView[642].

The method expects ChatRichView[642] to be formatted before they are called. It re-formats the changed message.

Parameters

Msg – the message whose image needs to be replaced. **Msg** must be present in Messages[643].

AText – new message text.

If **EditCount** >= 0, it is assigned to **Msg**.EditCount[666]. Otherwise, **Msg**.EditCount[666] is incremented.

Return value

True on success.

6.2.1.2.5 TRVChat.ClearMessages

Deletes all messages.

```
procedure ClearMessages;
```

The method clears ChatRichView[642] and Messages[643].

6.2.1.2.6 TRVChat.CopyToClipboard

Copies the selected chat content to the clipboard.

```
procedure DoCopyToClipboard(
  SelectCopiedMessages: Boolean = True);
```

If a single image is selected, ChatRichView[642] it is copied to the clipboard. Otherwise,

if ChatRichView[642] does not have a selection, and HighlightedMessage[642] is assigned, it is copied to the clipboard as a plain text. Otherwise,

If ChatRichView[642] contains a selected fragment, all messages at least partially included in it are copied to the clipboard as text (see GetMessagesAsText[651]). If **SelectCopiedMessages** = *True*, the method expands the selection to include all copied messages in full.

6.2.1.2.7 TRVChat.DeleteMessage

Deletes the specified message.

```
procedure DeleteMessage(AIndex: Integer);
procedure DeleteMessage(Msg: TRVChatMessage[665]);
```

The message can be specified by **AIndex** in Messages[643], or by a reference to an item in Messages[643].

The methods update both the content of ChatRichView[642] and deletes an item in Messages[643].

The method expects ChatRichView[642] to be formatted before they are called. It re-formats the changed fragment.

See also

- MarkMessageAsDeleted[653]

6.2.1.2.8 TRVChat.FindMessages

Returns a list of indexes of messages whose text contains the specified string.

```
function FindMessages(const Text: TRVUnicodeString[1099];
  AStartIndex: Integer = 0; AMaxCount: Integer = 0;
  CaseSensitive: Boolean = False): TRVIntegerList[1036];
```

Parameters

Text – text to find in messages.

The search starts from Messages[643] [**AStartIndex**].

If **AMaxCount** > 0, the search returns at most AMaxCount messages. If **AMaxCount** <= 0, the search returns all found messages.

CaseSensitive specifies whether the search is case-sensitive.

Return value

A list of the found messages, or **nil** if none are found.

6.2.1.2.9 TRVChat.GetAutoQuote, GetAutoQuoteFromItemNo

Returns a quote from the message as a text string. The quote can be limited to a specified number of characters.

```
function GetAutoQuoteFromItemNo(
  ItemNo: Integer): TRVUnicodeString[1099];

function GetAutoQuoteFromItemNo(
  ItemNo, MaxLength: Integer): TRVUnicodeString[1099];

function GetAutoQuote(
  Msg: TRVChatMessage[665]): TRVUnicodeString[1099];

function GetAutoQuote(
  Msg: TRVChatMessage[665]; MaxLength: Integer):
  TRVUnicodeString[1099];
```

The message is specified either by **Msg** parameter (a reference to one of the items in the Messages[643] collection) or by the **ItemNo** parameter (an item index in the ChatRichView[642] document – an index of any item belonging to the message can be used).

The length of the returned value is limited either by the **MaxLength** parameter (for methods that have it) or by the Quote[643].MaxAutoQuoteLength[680] property (for other methods).

6.2.1.2.10 TRVChat.GetFirstVisibleMessage

Returns the fist message visible in the client area of ChatRichView[642] (or nil, if the chat is empty).

```
function GetFirstVisibleMessage: TRVChatMessage[665];
```

6.2.1.2.11 TRVChat.GetImageAtCursor

Returns the image in ChatRichView[642] below the mouse pointer (or nil, if none)

```
function GetImageAtCursor: TRVGraphic[1032];
```

6.2.1.2.12 TRVChat.GetMessage, GetMessageAndImage

Returns the message by the item index in the ChatRichView[642] document.

```
function GetMessage(ItemNo: Integer): TRVChatMessage[665];

procedure GetMessageAndImage(ItemNo: Integer;
  out Msg: TRVChatMessage[665];  out Image: TRVGraphic[1032];
  out ImageText: TRVUnicodeString[1099]);
```

Input parameter:

ItemNo – the item index in the ChatRichView[642] document (an index of any item belonging to the message can be used).

Output:

Msg (or return value) – the message containing an item with the specified index, or *nil* if the specified item does not belong to a message.

Image – the image attached to this message (or *nil* if the message does not have an attached image)

ImageText – the text string associated with **Image**.

The opposite method to **GetMessage** is GetMessageItemNo[650].

6.2.1.2.13 TRVChat.GetMessageAt*

Returns the message at the specified coordinates (or *nil*, if none).

```
function GetMessageAtScreenCoords(
  const Pt: TRVCoordPoint[1064]): TRVChatMessage[665];

function GetMessageAtClientCoords(
  const Pt: TRVCoordPoint[1064]): TRVChatMessage[665];

function GetMessageAtDocumentCoords(
  const Pt: TRVCoordPoint[1064]): TRVChatMessage[665];

function GetMessageAtCursor: TRVChatMessage[665];
```

Pt parameter is a pont:

- For **GetMessageAtClientCoords**: in client coordinates of ChatRichView[642] (the origin is at the top left corner of ChatRichView[642] window)

- For **GetMessageAtDocumentCoords**: in document coordinates of ChatRichView[642] (the origin is at the top left corner of ChatRichView[642] scrollable area)
- For **GetMessageAtScreenCoords**: in desktop coordinates.

GetMessageAtScreenCoords returns the message below the mouse pointer.

6.2.1.2.14 TRVChat.GetMessageImage, GetMessageImageEx

Returns an image attached to the specified message.

```
function GetMessageImage(
  Msg: TRVChatMessage[665]): TRVGraphic[1032];

function GetMessageImageEx(
  Msg: TRVChatMessage[665];  out Image: TRVGraphic[1032];
  out ImageText: TRVUnicodeString[1099]): Boolean;
```

Input parameter:

Msg – message, a reference to an item in Messages[643].

Output:

Image (or return value) – an image attached to this message (or *nil* if the message does not have an attached image)

ImageText – a text string associated with **Image**.

6.2.1.2.15 TRVChat.GetMessageItemNo, GetMessageTextItemNo

GetMessageItemNo returns the index of the first item in the ChatRichView[642] document that represents the given message.

GetMessageItemTextNo returns the index of the first item in the ChatRichView[642] document that represents text of the given message.

```
function GetMessageItemNo(Msg: TRVChatMessage[665];
  StartItemNo: Integer = 0): Integer;

function GetMessageItemNo(MessageIndex: Integer;
  StartItemNo: Integer = 0): Integer;

function GetMessageTextItemNo(Msg: TRVChatMessage[665];
  StartItemNo: Integer = 0): Integer;

function GetMessageTextItemNo(MessageIndex: Integer;
  StartItemNo: Integer = 0): Integer;
```

Parameters

The message is specified either by **Msg** parameter (a reference to one of the items in the Messages[643] collection) or by the **MessageIndex** parameter (an index of item in Messages)[643].

The component searches for the message in ChatRichView[642], enumerating items starting from **StartItemNo**. By default, **StartItemNo** = 0, so it searches for this message from the beginning. If

you know an index of item of one of previous messages, you can pass it in StartItemNo parameter to increase efficiency. If the message is not found, the methods return -1.

Each message is represented by a single paragraph. It may have the following sections (any of them are optional):

- user name
- quote / attached files / image
- message text

Each message may have line breaks, but cannot have paragraph breaks.

GetMessageItemTextNo returns the index of the first item in the message text section, or -1 if the message does not have text.

The opposite method to **GetMessageItemNo** is GetMessage[649].

6.2.1.2.16 TRVChat.GetMessagesAsText

Returns a text representation of the specified range of messages.

```
function GetMessagesAsText(FromIndex,
  ToIndex: Integer): TRVUnicodeString[1099];
```

Parameters

FromIndex..**ToIndex** – a range of messages, indexes in Messages[643].

Return value

Returns a text representation of the specified range of messages.

See also

- CopyToClipboard[647]

6.2.1.2.17 TRVChat.GetSelectedMessage, GetRangeOfSelectedMessages

The methods return message(s) that contains a selected fragment in ChatRichView[642].

```
function GetSelectedMessage: TRVChatMessage[665];

function GetRangeOfSelectedMessages(
  out FirstIndex, LastIndex: Integer): Boolean;
```

Parameters

GetSelectedMessage returns a message that contains selected items. It returns *nil* if there is no selection in ChatRichView[642], if the selection is outside a message (for example, on a date), or if the selection starts in one message and ends in another.

GetRangeOfSelectedMessages returns the range of messages that are at least partially selected, as indexes in Messages[643] collection (or -1, if messages are not selected).

Do not confuse **GetSelectedMessages** with HighlightedMessage[642]. **GetSelectedMessages** returns the message containing the selected fragment, while HighlightedMessage[642] is set manually by the programmer.

6.2.1.2.18 TRVChat.GetSelectedTextForQuote, HasSelectedTextForQuote

The methods return information for quoting a message.

```
function GetSelectedTextForQuote(
  out Msg: TRVChatMessage[665];
  out Text: TRVUnicodeString[1099]): Boolean;

function HasSelectedTextForQuote: Boolean;
```

Parameters

HasSelectedTextForQuote returns *True* if ChatRichView[642] contains a fragment suitable for quoting; that is, there is a selected fragment entirely within the text of a single message.

GetSelectedTextForQuote returns this message along with the selected text (or **nil** if no text fragment of a single message is selected)

6.2.1.2.19 TRVChat.InsertMessage

The methods insert a new message at the position specified by **Index**. Two methods with the same functionality, differing only in how parameters are passed – directly or via a record.

```
function InsertMessage(AIndex: Integer;
  AUser: TRVChatUser[681];
  const AText: TRVUnicodeString[1099];
  ADateTime: TDateTime = 0;
  Extra: TRVChatMessageExtra[670] = nil;
  AState: TRVChatMessageState[1062] = rvcmsDefault):
  TRVChatMessage[665];

function InsertMessage(AIndex: Integer;
  const Data: TRVChatMessageInsertionInfo[675]):
  TRVChatMessage[665];
```

These methods are rarely used, since messages are usually added with AddMessage[645].

Possible usage scenarios: a chat client receives messages in a different order than they were written and wants to arrange them by time of writing. Potential Messages[643][] fields that can be used to determine the order of messages are DateTime[666] and GlobalIndex[667].

The methods update both the content of ChatRichView[642] and insert an item in Messages[643].

The methods expect ChatRichView[642] to be formatted before they are called. They format the added message.

Parameters

AIndex specifies the position of message insertion, in range from 0 to Messages[643].Count.

AUser – a reference to the user (the author of this message). This is a reference to one of items in Users[644].

AText – message text. If (**Extra** = *nil*) or (Extra is TRVChatMessageExtraQuote), **AText** cannot be empty. **AText** can be multi-line.

ADateTime – message time. If 0, the current time is used.

Extra – additional data attached to the message: files, an image, or a quote. It may be an object the following classes:

- TRVChatMessageExtraFiles[670] to attach files
- TRVChatMessageExtraImage[672] to attach an image
- TRVChatMessageExtraQuote[673] to add a quotation
- *nil* to attach nothing.

If **Extra**.AutoDestroy[670] = *True*, this method frees **Extra** (if the message was added successfully)

AState – message status.

Return value

A reference to the appended message on success, *nil* otherwise.

6.2.1.2.20 TRVChat.MarkMessageAsDeleted

Marks the message as deleted. Such a message contains nothing except the system text indicating that it was deleted.

```
procedure MarkMessageAsDeleted(
  AIndex: Integer);

procedure MarkMessageAsDeleted(
  Msg: TRVChatMessage[665]);
```

The message can be specified by **AIndex** in Messages[643], or by a reference to an item in Messages[643].

The methods update both the content of ChatRichView[642] and changes the message's State[667] to *rvcmsDeleted*.

See also

- DeleteMessage[647]

6.2.1.2.21 TRVChat.MoveMessage

Moves the specified message to the new position.

```
function MoveMessage(AIndexFrom,
  AIndexTo: Integer): Boolean;
```

The method reorder Messages[643]: moves Messages[643][**AIndexFrom**] to the position **AIndexTo**.

This method is rarely used, since messages are usually added with AddMessage[645] and not reordered.

The methods update both the content of ChatRichView[642] and deletes an item in Messages[643].

The method expects ChatRichView[642] to be formatted before they are called. It re-formats the changed fragment.

6.2.1.2.22 TRVChat.ScrollToMessage

Scrolls ChatRichView[642] to show the specified message.

```
procedure ScrollToMessage(
  AIndex: Integer);

procedure ScrollToMessage(
  Msg: TRVChatMessage[665]);
```

The message can be specified by **AIndex** in Messages[643], or by a reference to an item in Messages[643].

6.2.1.2.23 TRVChat.UpdateTextStyles, UpdateParaStyles

The method update properties of styles (which were added by AddStyles[654]) according to TRVChat properties.

```
procedure UpdateTextStyles;
procedure UpdateParaStyles;
```

Use these methods if you have changed the values of chat properties that affect text and/or paragraph styles. Then call ChatRichView[642].Reformat[339].

6.2.1.3 Events

In TRVChat

- OnClickDate[654]
- OnClickFile[654]
- OnClickImage[655]
- OnClickLink[655]
- OnClickUser[655]
- OnHighlightedMessageChange[655]

6.2.1.3.1 TRVChat.OnClickDate

Occurs while the user clicks on a date before messages.

```
type
  TRVClickDateEvent = procedure (Sender: TRVChat[640];
    ADate: TDateTime) of object;

property OnClickDate: TRVClickDateEvent;
```

6.2.1.3.2 TRVChat.OnClickFile

Occurs while the user clicks on a file attached to a message.

```
type
  TRVClickFileEvent = procedure (Sender: TRVChat[640];
    Msg: TRVChatMessage[665]; FileIndex: Integer)
    of object;

property OnClickFile: TRVClickFileEvent;
```

You can get information about the clicked file using **Msg**.Files [668][FileIndex].

6.2.1.3.3 TRVChat.OnClickImage

Occurs while the user clicks on an image attached to a message.

```
type
  TRVClickImageEvent = procedure (Sender: TRVChat[640];
    Msg: TRVChatMessage[665]; Image: TRVGraphic[1032];
    const ImageText: TRVUnicodeString[1099]) of object;

property OnClickImage: TRVClickImageEvent;
```

Msg – message containing the image, a reference to an item in Messages [643].

Image – an image attached to this message.

ImageText – a text string associated with **Image**.

6.2.1.3.4 TRVChat.OnClickLink

Occurs while the user clicks on a hyperlink in a message text.

```
type
  TRVClickLinkEvent = procedure (Sender: TRVChat[640];
    RVData: TCustomRVFormattedData[1014];
    ItemNo: Integer) of object;

property OnClickLink: TRVClickLinkEvent;
```

RVData and **ItemNo** define location of the hypertext item. You can use **RVData**.GetItemTextW [310] and GetItemTag [310] methods.

The chat does not use TRichView [216]'s standard hypertext functionality. Instead, it handles clicks on document items itself.

6.2.1.3.5 TRVChat.OnClickUser

Occurs while the user clicks on a user name in chat.

```
type
  TRVClickUserEvent = procedure (Sender: TRVChat[640];
    User: TRVChatUser[681]) of object;

property OnClickUser: TRVClickUserEvent;
```

User is a reference to an item in Users [644].

6.2.1.3.6 TRVChat.OnHighlightedMessageChange

Occurs when the value of HighlightedMessage [642] is changed.

```
property OnHighlightedMessageChange: TNotifyEvent;
```

This can happen when assigning a new value to the property or when deleting a message using the TRVChat [640] methods.

6.2.1.4 Related classes and records

Classes of TRVChat[640] Properties

- TRVChatCalloutProperties[657] is a type of TRVChat.Callout[641] property. It contains properties that define the callout bubble sizes, margins and padding, corner radii, arrow types and sizes, and shadow.
- TRVChatDateProperties[660] is a type of TRVChat.DateLine[642] property. It contains properties that define the color, margins, padding, and format for displaying dates of message groups.
- TRVChatFileIconsProperties[662] is a type of TRVChat.FileIcons[642] property. It contains properties that define the attributes of file icons attached to messages — both the icons drawn automatically by the chat and the images that can be provided instead of them.
- TRVChatFonts[664] is a type of TRVChat.Fonts[642]. This class contains properties defining fonts used in chat.
- TRVChatMessageCollection[668] is a type of TRVChat.Messages[643]. This class represents a collection of chat messages. The item's class is TRVChatMessage[665].
- TRVChatMessageStatusLine[676] is a type of TRVChat.MessageStatusLine[643]. This class contains properties that define the size, position, and content of the message status line. The status line can include the time, an edited marker, and an icon indicating the message status.
- TRVChatQuoteProperties[679] is a type of TRVChat.Quote[643]. This class contains properties defining how quotes are displayed in messages.
- TRVChatUITexts[680] is a type of TRVChat.Texts[643]. This class contains properties defining texts used in chat.
- TRVChatUserCollection[683] is a type of TRVChat.Users[644]. This class represents a collection of chat users (authors of messages). The item's class is TRVChatUser[681].
- TRVChatUserPictureProperties[684] is a type of TRVChat.UserPicture[644]. This class contains properties defining attributes of speakers' pictures (avatars).

Classes of TRVChatMessage[665] Properties

- TRVChatMessageQuote[675] is a type of TRVChatMessage.Quote[667]. This class contains information about a quote from another message attached to the given message.
- TRVChatFileCollection[661] is a type of TRVChatMessage.Files[668]. This class contains information about a files attached to the given message. The item's class is TRVChatFileItem[663].

Classes used to create a message

- TRVChatMessageExtraFiles[670] is used to attach files to a new message.
- TRVChatMessageExtraImage[672] is used to attach an image to a new message.
- TRVChatMessageExtraQuote[673] is used to add a quote to a new message.

Records

TRVChatMessageInsertionInfo[675] contains information for message creation.

6.2.1.4.1 TRVChatCalloutProperties

This is a class of Callout[641] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatCalloutProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties that define the callout bubble sizes, margins and padding, corner radii, arrow types and sizes, and shadow.

6.2.1.4.1.1 Properties

In TRVChatCalloutProperties

- ArrowHeight[657]
- ArrowWidth[657]
- CalloutType[658]
- CornerRadiusLarge[658]
- CornerRadiusSmall[658]
- Margin[658]
- MaxWidth[658]
- MinOppositeMargin[659]
- MinWidth[658]
- Padding[659]
- ShadowOpacity[659]
- Spacing[659]

These properties define size of callout arrow (that points to the speaker's avatar).

```
property ArrowHeight: TRVPixel96Length[1085];
property ArrowWidth: TRVPixel96Length[1085];
```

Zero or positive values are allowed.

These properties are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default values

- ArrowHeight: 10
- ArrowWidth: 10

Defines the appearance of the callout arrow.

```
type
  TRVCalloutType = (rvctCurved, rvctStraight);

property CalloutType: TRVCalloutType;
```

Value	Meaning
rvctCurved	the arrow flows smoothly into the callout.
rvipRightMargin	the arrow is drawn with straight lines

Default value

rvctCurved

The properties define the corner rounding radii of the callout bubble. A smaller radius is used when bubbles from the same speaker appear consecutively.

```
property CornerRadiusSmall: TRVPixel96Length[1085];
property CornerRadiusLarge: TRVPixel96Length[1085];
```

Zero or positive values are allowed.

These properties are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default values

- CornerRadiusSmall: 5
- CornerRadiusLarge: 10

Vertical margin for callout bubbles.

```
property Margin: TRVPixel96Length[1085];
```

This property defines external margins above and below bubbles.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

3

The properties define the minimum and maximum callout bubble width.

```
property MaxWidth: TRVPixel96Length[1085];
property MinWidth: TRVPixel96Length[1085];
```

Several consecutive bubbles from the same speaker for messages sent sequentially on the same day form a group. **Spacing** is added before this group, in addition to regular Margin[658].

These properties are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

- MinWidth: 100
- MaxWidth: 440

The minimum margin between the callout bubble edge and the chat edge on the side opposite to the speaker's avatar.

```
property MinOppositeMargin: TRVPixel96Length[1085];
```

This property defines external margins above and below bubbles.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

3

Padding around the callout bubble.

```
property Padding: TRVPixel96Length[1085];
```

This property defines padding (internal margins) around contents of bubbles.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

10

Opacity of shadows of callout bubbles.

```
property ShadowOpacity: TRVOpacity[1082];
```

Set to 0 to hide shadows.

Default value

35000 (35%)

Additional spacing before the first callout bubble in a group.

```
property Spacing: TRVPixel96Length[1085];
```

Several consecutive bubbles from the same speaker for messages sent sequentially on the same day form a group. **Spacing** is added before this group, in addition to regular Margin[658].

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

2

6.2.1.4.2 TRVChatDateProperties

This is a class of DateLine[642] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatDateProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties that define the color, margins, padding, and format for displaying dates of message groups.

See also
RVChat.Fonts[642].DateFont.

6.2.1.4.2.1 Properties

In TRVChatDateProperties

- BackColor[660]
- DateFormatNew[660]
- DateFormatOld[660]
- MarginAbove[661]
- MarginBelow[661]
- PaddingHorizontal[661]
- PaddingVertical[661]

Background color of rounded rectangles containing dates.

```
property BackColor: TRVColor[1063];
```

Default value

rvclGray[1105]

Format strings used by FormatDateTime function to display dates of groups of messages.

```
property DateFormatNew: String;
property DateFormatOld: String;
```

DateFormatNew is used for dates from the current year, and **DateFormatOld** is used for older dates.

Default value

- DateFormatNew: 'mmmm d'
- DateFormatOld: 'mmmm d, yyyy'

Vertical margins for rounded rectangles with dates.

```
property MarginAbove: TRVPixel96Length[1085];
property MarginBelow: TRVPixel96Length[1085];
```

These properties define external margins above and below rounded rectangles with dates.

These properties are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

- MarginAbove: 5
- MarginBelow: 3

Padding for rounded rectangles with dates.

```
property PaddingVertical: TRVPixel96Length[1085];
property PaddingHorizontal: TRVPixel96Length[1085];
```

These properties define padding (internal margins) around a date text.

These properties are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

- PaddingVertical: 6
- PaddingHorizontal: 15

6.2.1.4.3 TRVChatFileCollection

This is a class of Files[668] property of TRVChatMessage[665]. A collection of TRVChatFileItem[663] items.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatFileCollection = class (TOwnedCollection)
```

Hierarchy

TObject
TPersistent
TCollection
TOwnedCollection

Description

This class represents a collection of attached files. The item's class is TRVChatFileItem[663].

6.2.1.4.3.1 Properties

In TRVChatFileCollection

- Items[662]

Items of this collection.

```
property Items[Index: Integer]: TRVChatFileItem[663];
```

Each item represents a single attached file.

6.2.1.4.4 TRVChatFileIconsProperties

This is a class of FileIcons[642] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatFileIconsProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties that define the attributes of file icons attached to messages — both the icons drawn automatically by the chat and the images that can be provided instead of them.

6.2.1.4.4.1 Properties

In TRVChatFileIconsProperties

- PictureArchive[663]
- PictureAudio[663]
- PictureDocument[663]
- PictureDownload[663]
- PictureImage[663]
- PictureVideo[663]
- IconColor[662]
- IconPadding[662]
- IconSize[662]
- ScalePictures[663]

These properties allow you to set the icons for files attached to messages.

```
property IconSize: TRVPixel96Length[1085];
property IconColor: TRVColor[1063];
property IconPadding: TRVPixel96Length[1085];
```

An icon depends on the type of the attached file, which is determined by its extension. These properties define the appearance of the icons that the chat draws automatically. Such icons are used if no images are provided in the Picture*[663] properties.

The icons appear as symbols inside a circle. **IconSize** defines the icon's height and width, that is, the circle's diameter. The size of the symbol inside the circle is **IconSize** - **IconPadding** * 2. The

symbol is drawn using **IconColor**, and the circle is filled with the color used to display the speaker's name.

If ScalePictures [663] = *True*, then **IconSize** also affects the images specified in the Picture* [663] properties.

IconSize and **IconPadding** are measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default values

- IconColor: *rvclWhite* [1105]
- IconSize: 50
- IconPadding: 10

These properties allow you to set the icons for files attached to messages.

```
property PictureArchive: TRVPicture[1037];
property PictureAudio: TRVPicture[1037];
property PictureDocument: TRVPicture[1037];
property PictureDownload: TRVPicture[1037];
property PictureImage: TRVPicture[1037];
property PictureVideo: TRVPicture[1037];
```

An icon depends on the type of the attached file, which is determined by its extension. By default, the chat draws the icons automatically. In these properties, you can override the icons for several or all file types.

By default, these pictures are shown at their original size. If ScalePicture [663] = *True*, then they are scaled to IconSize [662] x IconSize.

Specifies whether the images set in the Picture* [663] properties will be scaled to IconSize [662].

```
property ScalePictures: Boolean;
```

Default value

False

6.2.1.4.5 TRVChatFileItem

This is a class of items of Files [668] collection property of TRVChatMessage [665].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatFileItem = class (TCollectionItem)
```

Hierarchy

TObject
TPersistent
TCollectionItem

Description

This class represents a single file attached to a message. This is an item of TRVChatFileCollection[661].

6.2.1.4.5.1 Properties

In TRVChatFileItem

- DisplayName[664]
- FileName[664]
- FileSize[664]
- FileType[664]

The file name as it will be displayed in the chat window.

```
property DisplayName: TRVUnicodeString[1099];
```

The file location.

```
property FileName: TRVUnicodeString[1099];
```

This can be a file name or any string indicating its location. For example, if you received a message from the server but the file has not yet been downloaded, this property may contain a command to retrieve the file from the server.

The file size, in bytes.

```
property FileSize: Int64;
```

File type.

```
property FileType: TRVChatFileType[1061];
```

The chat displays the file type with a corresponding icon.

See also

- TRVChat[640].FileIcons[642]

6.2.1.4.6 TRVChatFonts

This is a class of Fonts[642] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatFonts = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties defining fonts used in chat.
All properties are of the TRVFont[1032] type.

- MessageFont: font for messages
- UserNameFont: font for user names
- QuoteFont: font for quoted text
- DateFont: font for dates of message groups
- TimeFont: font for message time
- FileNameFont: font for names of files attached to messages
- FileInfoFont: for for information about files attached to messages
- DeletedMessageFont: font for information about message that was marked as deleted.

Default values

- MessageFont: (10, [], *rvclBlack*[1105])
- UserNameFont: (10, [*rvfsBold*], *rvclBlack**)
- QuoteFont: (10, [*rvfsItalic*], *rvclBlack*)
- DateFont: (10, [*rvfsBold*], *rvclWhite*)
- TimeFont: (9, [], *rvclGray***)
- FileNameFont: (10, [*rvfsBold*], *rvclBlack*)
- FileInfoFont: f(9, [], *rvclGray*)
- DeletedMessageFont: (10, [*rvfsItalic*], *rvclGray*)

* **UserNameFont**.Color is ignored; Chat.Users[644][].NameColor[682] is used instead.

** If Chat.MessageStatusLine[643].StatusLineLocation[678] = *rvsllBelowInside,* Chat.Users[644][].MessageInfoColor[682] overrides **TimeFont**.Color.

All the font have Name = 'Segoe UI'. You can set the font name for all these fonts using the method

```
procedure SetFontName(const FontName: TRVFontName);
```

You can also use the method

```
procedure SetDefautlFontName;
```

to assign RVDefaultLoadProperties[1119].DefaultFontName[1026] to all fonts. It is especially useful in cross-platform applications.

These properties are used by Chat.AddStyles[646] to append text styles.

If you change the fonts while the chat is already displayed, call Chat.UpdateTextStyles[654] to apply the new values.

See also:

- TRVChat.LinkColor[643]

6.2.1.4.7 TRVChatMessage

This is a class of items of Messages[643] collection property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessage = class (TCollectionItem)
```

Hierarchy

TObject
TPersistent
TCollectionItem

Description

This class represents a chat message. This is an item of TRVChatMessageCollection[668].

All operations with messages must be performed using the TRVChat[640] methods. The only exception is changing the message state[667] (to new, sent, read, or failed).

6.2.1.4.7.1 Properties

In TRVChatMessage

DateTime[666]
EditCount[666]
Files[668]
GlobalIndex[667]
Id[666]
Quote[667]
State[667]
Text[667]
User[667]

Identifier of the chat message.

```
property Id: TRVChatItemId[1061];
```

This property is not used by the chat. You can assign and use it as you see fit.

You can search for a collection item by its **Id** using the FindById and FindItemById[669] methods.

Default value

0

Date and time of the message.

```
property DateTime: TDateTime;
```

The property stores how many times the message has been edited.

```
property EditCount: Integer;
```

Initial value

0 (not edited)

Provides an additional way to order the messages.

```
property GlobalIndex: Integer;
```

This property is not used by the chat itself but allows you to assign additional numbering to messages. For example, such numbering could be maintained by the chat server in the order messages are received from clients, and the server can send messages to clients including their global index.

You can search for a collection item by its **GlobalIndex** using the FindByGlobalIndex and FindItemByGlobalIndex[669] methods.

Initial value

0

Message text.

```
property Text: TRVUnicodeString[1099];
```

Initial value

'' (empty string)

Message status.

```
property State: TRVChatMessageState[1062];
```

"New", "sent", "read", and "failed" states can be displayed using icons from a TImageList[679].

In deleted messages, content is replaced to Chat.Texts[643].DeletedMessage[680]

This is the only message property that can be modified directly (except for assigning *rvcmsDeleted*). All other operations with messages must be performed using the TRVChat[640] methods.

Initial value

rvcmsDefault

A reference to the user (the author of this message).

```
property User: TRVChatUser[681];
```

This is a reference to one of items in Chat.Users[644].

Initial value

rvcmsDefault

Contains information about a quote from another message attached to this message.

```
property Quote: TRVChatMessageQuote[675];
```

If this message has no attached quote, it is *nil*.

Contains information about files attached to this message.

```
property Files: TRVChatFileCollection[661];
```

If this message has no attached files, it is *nil*.

6.2.1.4.8 TRVChatMessageCollection

This is a class of Messages[643] property of TRVChat[640]. A collection of TRVChatMessage[665] items.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageCollection = class (TOwnedCollection)
```

Hierarchy

TObject
TPersistent
TCollection
TOwnedCollection

Description

This class represents a collection of chat messages. Each item represents a single message in the order it is displayed in the chat. The item's class is TRVChatMessage[665].

You should not add, delete, or modify the properties of this collection directly. All operations with messages must be performed using the chat's methods. The only exception is changing the message state[667] (to new, sent, read, or failed).

6.2.1.4.8.1 Properties

In TRVChatMessageCollection

- Items[668]

Items of this collection.

```
property Items[Index: Integer]: TRVChatMessage[665];
```

Each item represents a single message in the order it is displayed in the chat.

6.2.1.4.8.2 Methods

In TRVChatMessageCollection

FindByDate[669]
FindByGlobalIndex[669]
FindById[669]
FindInsertionPlace[670]
FindItemByDate[669]

FindItemByGlobalIndex[669]
FindItemById[669]

Returns the first message written on the specified date or later.

```
function FindByDate(
  ADate: TDate): Integer;

function FindItemByDate(
  ADate: TDate): TRVChatMessage[665];
```

The methods assume that all messages are sorted in ascending order by the DateTime[666] property.

If found, **FindItemByDate** returns the item; **FindByDate** returns its index (zero-based).

If not found, **FindItemByDate** returns *nil*; **FindByDate** returns -1.

See also:

- FindByGlobalIndex, FindItemByGlobalIndex[669]
- FindById, FindItemById[669]

These methods search for a collection item by its GlobalIndex[667].

```
function FindByGlobalIndex(
  GlobalIndex: Integer): Integer;

function FindItemByGlobalIndex(
  GlobalIndex: Integer): TRVChatMessage[665];
```

If found, **FindItemByGlobalIndex** returns this item; **FindByGlobalIndex** returns its index (zero-based).

If not found, **FindItemByGlobalIndex** returns *nil*; **FindByGlobalIndex** returns -1.

See also:

- FindInsertionPlace[670]
- FindById, FindItemById[669]
- FindByDate, FindItemByDate[669]

These methods search for a collection item by its Id[666].

```
function FindById(
  const Id: TRVChatItemId[1061]): Integer;

function FindItemById(
  const Id: TRVChatItemId[1061]): TRVChatMessage[665];
```

If found, **FindItemById** returns this item; **FindById** returns its index (zero-based).

If not found, **FindItemById** returns *nil*; **FindById** returns -1.

See also:

- FindByGlobalIndex, FindItemByGlobalIndex[669]
- FindByDate, FindItemByDate[669]

Returns the index in the collection where a new message should be inserted by GlobalIndex[667].

```
function FindInsertionPlace(GlobalIndex: Integer): Integer;
```

This method is rarely used. It assumes that you are receiving messages from some server that maintains its own message counter, and you may receive messages from it in a different order than they should be displayed. In this case, this method indicates where the new message with the given **GlobalIndex** should be inserted. It assumes that all previously added messages are sorted in ascending order by the GlobalIndex[667] property.

See also:

- FindByGlobalIndex, FindItemByGlobalIndex[669]

6.2.1.4.9 TRVChatMessageExtra

This is a class is used to attach additional information to a chat message.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageExtra = class
```

Hierarchy

TObject

Description

This class is not used directly.
Use the following inherited classes:

- TRVChatMessageExtraFiles[670] to attach files
- TRVChatMessageExtraImage[672] to attach an image
- TRVChatMessageExtraQuote[673] to add a quotation.

6.2.1.4.9.1 Properties

In TRVChatMessageExtra

AutoDestroy[670]

Specifies whether the memory for this class object will be freed automatically when the message is added to the chat.

```
property AutoDestroy: Boolean;
```

Default value

True

6.2.1.4.10 TRVChatMessageExtraFiles

This is a class is used to attach files to a chat message.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageExtraFiles = class(TRVChatMessageExtra[670])
```

Hierarchy

TObject
TRVChatMessageExtra[670]

Description

Create this class, then call AddFile[671] for each file to attach.

6.2.1.4.10.1 Properties

In TRVChatMessageExtraFiles

Files[671]

Inherited from TRVChatMessageExtra[670]

AutoDestroy[670]

The files that will be attached to the message.

```
property Files: TRVChatFileCollection[661];
```

While you can use this collection to add files, it's recommended to use AddFile[671] instead. This method can retrieve the size of the file.

6.2.1.4.10.2 Methods

In TRVChatMessageExtraFiles

AddFile[671]
AssignDefaultFileTypes[672]
Create[672]

Adds a new file that will be attached to the message.

```
procedure AddFile(const ADisplayName,
  AFileName: TRVUnicodeString[1099];
  AFileSize: Int64 = -1);
```

Parameters

ADisplayName – the file name as it will be displayed in the chat window.

AFileName can be a file name or any string indicating its location. For example, if you received a message from the server but the file has not yet been downloaded, this property may contain a command to retrieve the file from the server.

AFileSize – file size in bytes. If **AFileName** contains a full local file name, you can pass -1 in this parameter. In this case the component will retrieve the file size itself.

The file will be added with the file type[1061] = *rvcftDownload*. You can modify it in Files collection, or call AssignDefaultFileTypes[672] after adding all files.

The method automatically determines file types[1061] by their extensions.

```
procedure AssignDefaultFileTypes;
```

Files with an unknown type are classified as *rvcftDocument*.

This method works for all items in the file collection, so it makes sense to call it after all files have been added[671].

Creates a new instance of object of this class.

```
constructor Create;
```

After creation, call AddFile[671] for each file.

6.2.1.4.11 TRVChatMessageExtraImage

This is a class is used to attach an image to a chat message.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageExtraImage = class(TRVChatMessageExtra[670])
```

Hierarchy

TObject
TRVChatMessageExtra[670]

Description

When creating this class, you specify:
- the image to insert
- an optional text associated with this image.

6.2.1.4.11.1 Properties

In TRVChatMessageExtraImage

Image[673]
ImageText[673]
MustAddCopyOfImage[673]

Inherited from TRVChatMessageExtra[670]

AutoDestroy[670]

Return the image that will be attached to a message.

```
property Image: TRVGraphic;
```

Return the text associated with the image that will be attached to a message.

```
property ImageText: TRVUnicodeString;
```

This property is not used by the chat; you can use it at your discretion.

A possible scenario: if an image has not yet been downloaded, you can insert a placeholder and store the address for downloading the image in this property. Later, you can handle a click on the image to start the download and then replace the placeholder with the downloaded image.

Specifies whether the image itself or its copy will be inserted into the chat.

```
property MustAddCopyOfImage: Boolean;
```

If the image itself is inserted, the chat will free the Image when it is no longer used.

If a copy is inserted, you must free the memory for the Image yourself.

6.2.1.4.11.2 Methods

In TRVChatMessageExtraImage

Create

Creates a new instance of object of this class.

```
constructor Create(AImage: TRVGraphic;
  AMustAddCopyOfImage: Boolean = True;
  AImageText: TRVUnicodeString = '');
```

Parameters:

AImage – the image to attach to the message.

AMustAddCopyOfImage specifies whether the image itself or its copy will be inserted into the chat.

AImage – additional text associated with the image.

See the properties of this class for additional details.

6.2.1.4.12 TRVChatMessageExtraQuote

This is a class is used to attach a quote to a chat message.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageExtraQuote = class(TRVChatMessageExtra)
```

Hierarchy

TObject
TRVChatMessageExtra

Description

When creating this class, you specify:
- the message to quote
- text to quote (if you do not quote the whole message)

6.2.1.4.12.1 Properties

In TRVChatMessageExtraQuote

Msg [674]
Text [674]
User [674]

Inherited from TRVChatMessageExtra [670]

AutoDestroy [670]

A reference to the quoted message.

```
property Msg: TRVChatMessage[665];
```

This is a reference to an item in Chat.Messages [643].

Quoted text.

```
property Text: TRVUnicodeString[1099];
```

If Text is empty, this is a quote of the whole message.

A reference to the user, the author of the quoted message.

```
property User: TRVChatUser[681];
```

This is a reference to an item in Chat.Users [644].

Normally, this property is equal to Msg [676].User [667]. However, if you add a quote to the deleted message, the user must be specified.

6.2.1.4.12.2 Methods

In TRVChatMessageExtraQuote

Create [674]

Creates a new instance of object of this class.

```
constructor Create(AMsg: TRVChatMessage[665];
  const AText: TRVUnicodeString[1099] = '';
  AUser: TRVChatUser[681] = nil);
```

Parameters:

AMsg – the message that will be quoted. Maybe nil if you insert a quote to a deleted message.

AText – quotation; if empty, the whole message is quoted.

AUser – the author of AMsg. If nil, **AMsg**.User[667] will be used.

At least one of **AMsg** and **AUser** must be not *nil*.

6.2.1.4.13 TRVChatMessageInsertionInfo

A record containing information about a chat message.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
type
  TRVChatMessageInsertionInfo = record
    User: TRVChatUser[681];
    Text: TRVUnicodeString[1099];
    DateTime: TDateTime;
    Extra: TRVChatMessageExtra;
    State: TRVChatMessageState[1062];
    Id: TRVAnsiString[1056];
    GlobalIndex, EditCount: Integer;
  end;
```

See Chat.AddMessage[645] for an explanation of the structure's fields.

6.2.1.4.14 TRVChatMessageQuote

This is a type of TRVChatMessage[665].Quote[667] property.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatMessageQuote = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

An object of this class contains information about a quote from another message attached to the given message.

6.2.1.4.14.1 Properties

In TRVChatMessageQuote

Msg[676]
QuoteType[676]
Text[676]
User[676]

A reference to the quoted message.

```
property Msg: TRVChatMessage[665];
```

This is a reference to an item in Chat.Messages[643].

If the quoted message has been deleted, this property returns *nil*.

Specifies the quote type.

```
property QuoteType: TRVChatQuoteType[1062];
```

It can be either a quote of the entire message or a fragment of its text[676].

Quoted text.

```
property Text: TRVUnicodeString[1099];
```

This text is used if QuoteType[676] = *rvcqtQuotation*.

A reference to the user, the author of the quoted message.

```
property User: TRVChatUser[681];
```

This is a reference to an item in Chat.Users[644].

Normally, this property is equal to Msg[676].User[667]. However, if the quoted message has been deleted, this property retains a reference to its author.

6.2.1.4.15 TRVChatMessageStatusLine

This is a class of MessageStatusLine[643] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
MessageStatusLine = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties that define the size, position, and content of the message status line. The status line can include the time, an edited marker, and an icon indicating the message status.

6.2.1.4.15.1 Properties

In TRVChatQuoteProperties

- FailedImageIndex 679
- Images 677
- ImagesHeight 677 [FMX]
- ImagesWidth 677 [FMX]
- MinStatusLineHeight 677
- NewImageIndex 679
- ReadImageIndex 679
- SentImageIndex 679
- StatusLineContent 678
- StatusLineLocation 678

Defines the reference to a TCustomImageList list of images to be used to draw images indicating states of messages.

```
property Images: TCustomImageList;
```

The images indicate the message status (sent, read, etc.).

See also:

- FailedImageIndex, NewImageIndex, ReadImageIndex, SentImageIndex 679

The properties define the size for drawing Images 677.

```
property ImagesWidth: Integer;
property ImagesHeight: Integer;
```

The images indicate the message status (sent, read, etc.).

Default value

16

See also:

- FailedImageIndex, NewImageIndex, ReadImageIndex, SentImageIndex 679

Specifies the minimal status line height for messages (unless no content is specified 678)

```
property MinStatusLineHeight: TRVPixel96Length 1085;
```

Status lines are tall enough to fit both text and images. This property allows you to make them even taller.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

16

Specifies what content will be displayed in the status line of each message.

```
type
  TRVChatStatusLineItem = (
    rvsliTime, rvsliState, rvsliEdited
  );
  TRVChatStatusLineItems = set of TRVChatStatusLineItem;

property StatusLineContent: TRVChatStatusLineItems;
```

Value	Meaning
rvsliTime	message time
rvsliState	image of message state [679]
rvsliEdited	text indicating that the message has been edited

Default value

[rvsliTime, rvsliState, rvsliEdited]

Specifies what content will be displayed in the status line of each message.

```
type
  TRVChatStatusLineLocation = (
    rvsllBelowInside, rvsllBelowOutside
  );

property StatusLineLocation: TRVChatStatusLineLocation;
```

Value	Meaning
rvsllBelowInside	Inside a callout bubble, below a message content
rvsllBelowOutside	Below a callout bubble

Default value

rvsllBelowInside

The properties declare the zero-based indexes in Images[677]. The images indicate the message status (sent, read, etc.).

```
property NewImageIndex: Integer;
property SentImageIndex: Integer;
property ReadImageIndex: Integer;
property FailedImageIndex: Integer;
```

See the values of message's State[667].

Default value

-1

6.2.1.4.16 TRVChatQuoteProperties

This is a class of Quote[643] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatQuoteProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

Quoted text is displayed inside a rectangle with rounded corners. The properties of this rectangle are defined in this class.

6.2.1.4.16.1 Properties

In TRVChatQuoteProperties

- CompactQuotes[679]
- CornerRadius[680]
- MaxAutoQuoteLength[680]
- Padding[680]

Specifies whether the quote formatting is compact.

```
property CompactQuotes: Boolean;
```

Default value

False

Defines the corner rounding radius of the quote rectangle.

```
property CornerRadius: TRVPixel96Length[1085];
```

Zero or positive values are allowed.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

5

Specifies the maximum number of characters in a quote when quoting an entire message.

```
property MaxAutoQuoteLength: Integer;
```

For longer messages, the text is truncated and the removed part is replaced with an ellipsis. If the value is zero or negative, all messages are quoted in full.

This property only affects quotes having QuoteType[676] = *rvcqtMessage*.

Default value

25

See also:

- TRVChat[640].GetAutoQuote, GetAutoQuoteFromItemNo[648]

Padding around the quoted text.

```
property Padding: TRVPixel96Length[1085];
```

This property defines padding (internal margins) around quoted text.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

10

6.2.1.4.17 TRVChatUITexts

This is a class of Texts[643] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatUITexts = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties defining texts used in chat.
All properties are of the TRVUnicodeString[1099] type.

File size measure units:
- B: bytes ('B')
- KB: kilobytes ('KB')
- MB: megabytes ('MB')
- GB: gigabytes ('GB')

Other strings:
- Edited: shown for edited messages ('edited')
- DeletedMessage: shown for messages marked as deleted ('Deleted message')

6.2.1.4.18 TRVChatUser

This is a class of items of Users [644] collection property of TRVChat [640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatUser = class (TCollectionItem)
```

Hierarchy

TObject
TPersistent
TCollectionItem

Description

This class represents a chat user (authors of messages). This is an item of TRVChatUserCollection [683].

Each chat message is associated with one of the users (the one who wrote it), see TRVChatMessage [665].User [667].

6.2.1.4.18.1 Properties

In TRVChatUser

- FillColor [682]
- Id [682]
- IsRightSide [682]
- MessageInfoColor [682]
- Name [682]
- NameColor [682]
- Picture [683]
- QuoteFillColor [683]
- ShowNameInMessages [683]
- ShowPicture [683]

The fill color of the callout bubble.

```
property FillColor: TRVColor;
```

Default value

rvclWhite

Identifier of the chat user.

```
property Id: TRVChatItemId;
```

This property is not used by the chat. You can assign and use it as you see fit.

You can search for a collection item by its **Id** using the FindById and FindItemById methods.

Default value

0

Specifies whether this user's messages will be aligned to the left or right side of the chat window.

```
property IsRightSide: Boolean;
```

Default value

False

The text color of the status line in this user's messages (if the status line is inside the callout bubble).

```
property MessageInfoColor: TRVColor;
```

If Chat.MessageStatusLine.StatusLineLocation = *rvsllBelowInside,* this value overrides Chat.Fonts.TimeFont.Color. Otherwise, this property is ignored.

Default value

rvclGray

Name of the chat user.

```
property Name: TRVUnicodeString;
```

If ShowNameInMessages = *True*, the user's name is displayed in the first message of a group of messages written by them.

If you change a user's name, it will only affect new messages.

Default value

'User'

The text color of this user name.

```
property NameColor: TRVColor;
```

This color is also used in other places. For example, to draw the icons of files attached by this user and to automatically determine the background color of quotes from this user's messages.

Default value

rvclBlack

See also

- Chat.Fonts 642.UserNameFont 664

This user's picture (avatar).

```
property Picture: TRVPicture 1037;
```

If ShowPicture 683 = *True*, this user's picture is displayed next to this user's messages.

When displaying, the picture is scaled to Chat.UserPicture 644.Size 685 x Chat.UserPicture.Size.

The fill color of quotes and attached files.

```
property QuoteFillColor: TRVColor 1063;
```

This color is used:

- when another user quotes this user, or
- when this user attaches files to their message.

If this property is set to *rvclNone*, the colors of this user's quotes and files will be chosen automatically by the chat.

Default value

rvclNone 1105

Specifies whether this user's Name 682 will be added to this user's messages.

```
property ShowNameInMessages: Boolean;
```

Default value

True

Specifies whether this user's Picture 683 will be displayed next to this user's messages.

```
property ShowPicture: Boolean;
```

Default value

True

6.2.1.4.19 TRVChatUserCollection

This is a class of Users 644 property of TRVChat 640. A collection of TRVChatUser 681 items.

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatUserCollection = class (TOwnedCollection)
```

Hierarchy

TObject
TPersistent
TCollection
TOwnedCollection

Description

This class represents a collection of chat users (speakers). The item's class is TRVChatUser[681].

Each chat message is associated with one of the users (the one who wrote it).

When the chat is already populated, you can add new users to the collection and modify all user properties (except for the name, see below).

You cannot delete users who have messages in the chat.

If you change a user's name, it will only affect new messages.

6.2.1.4.19.1 Properties

In TRVChatUserCollection

- Items[684]

Items of this collection.

```
property Items[Index: Integer]: TRVChatUser[681];
```

Each item represents one chat user.

6.2.1.4.19.2 Methods

In TRVChatUserCollection

FindById[684]
FindItemById[684]

These methods search for a collection item by its Id[682].

```
function FindById(
  const Id: TRVChatItemId[1061]): Integer;

function FindItemById(
  const Id: TRVChatItemId[1061]): TRVChatUser[681];
```

If found, **FindItemById** returns this item; **FindById** returns its index (zero-based).

If not found, **FindItemById** returns *nil*; **FindById** returns -1.

6.2.1.4.20 TRVChatUserPictureProperties

This is a class of UserPicture[644] property of TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat.

Syntax

```
TRVChatUserPictureProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Description

This class contains properties that define the attributes of speakers' pictures (avatars).

6.2.1.4.20.1 Properties

In TRVChatUserPictureProperties

- Position [685]
- Size [685]

Specifies the vertical position of speakers' images relative to the group of callout bubbles.

```
type
  TRVChatUserPicturePosition = (
    rvcuppBottom, rvcuppTop, rvcuppNone
  );
property Position: TRVChatUserPicturePosition;
```

Value	Meaning
rvcuppBottom	Bottom of the last message in a group
rvcuppTop	Top of the first message in a group
rvcuppNone	No images

Default value

rvcuppBottom

Specifies the size of the speakers' images.

```
property Size: TRVPixel96Length [1085];
```

Positive values are allowed. Images are scaled to **Size** x **Size**.

This property is measured in pixels, where 1 pixel = 1/96 of an inch. The actual size is calculated based on the document's DPI value.

Default value

34

6.2.2 TRVDataSourceLink

A simple component that links data-aware controls inside TRichView [216], TRichViewEdit [469], or TSRichViewEdit component to a TDataSource component.

Unit RVDataSourceLink;

Syntax

```
TRVDataSourceLink = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

How to Use

This component automatically assigns DataSource[686] to data-aware controls when they are inserted in Editor[686].

It is especially useful when controls are loaded from RVF files, because TRichView cannot read DataSource property (and other properties pointing to components) from RVF.

Editor[686] can be a data-aware control by itself (TDBRichView[604], TDBRichViewEdit[616] or TDBSRichViewEdit from ScaleRichView). You can implement a master-detail view by linking the editor to a master data set, and controls in this editor to a detail data set.

6.2.2.1 Properties

In TRVDataSourceLink

- DataSource[686]
- Editor[686]

6.2.2.1.1 TRVDataSourceLink.DataSource

A TDataSource component for assigning to data-aware controls in Editor[686].

```
property DataSource: TDataSource;
```

TDataSource provides an interface between a dataset component and data-aware controls in Editor[686].

Value of this property is assigned automatically to DataSource properties of all data-aware controls when they are inserted in Editor[686] (if another DataSource is not already assigned to them).

When you assign a new value to this property, existing data-aware controls in Editor[686] are not affected; this value will be used only for controls that will be inserted in Editor[686] later. If you want to change DataSource of existing controls, call Execute[687].

6.2.2.1.2 TRVDataSourceLink.Editor

A TRichView or ScaleRichView component that contains (or will contain) data-aware controls.

```
property Editor: TCustomRVControl[869];
```

When a new data-aware control is inserted in **Editor**, this component assigns DataSource[686] to its DataSource property.

If you want to change DataSource of controls that are already in **Editor**, call Execute[687].

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

- TRichView[216]
- TRichViewEdit[616]

- TDBRichView 604
- TDBRichViewEdit 616
- TSRichViewEdit (from ScaleRichView)
- TDBSRichViewEdit (from ScaleRichView)

Only one TRVDataSourceLink 685 component can be linked an editor. If you assign an editor that already has another TRVDataSourceLink component assigned, an exception occurs.

ScaleRichView note: if a ScaleRichView component is assigned to this property, TRVDataSourceLink handles only controls in its main editor (RichViewEdit property), not in its headers, footers or notes.

6.2.2.2 Methods

In TRVDataSourceLink

Execute 687

6.2.2.2.1 TRVDataSourceLink.Execute

Assigns DataSource 686 to all data-aware controls in Editor 686.

```
procedure Execute(OnlyToUndefined: Boolean);
```

Normally, you do not need to call this method, because DataSource 686 is assigned to to all data-aware controls in Editor 686 when they are inserted (for example, when loaded from RVF).

This method may be useful if you want to change DataSource in controls that were already inserted in Editor 686.

Parameters:

If **OnlyToUndefined** = *True*, DataSource will be assigned only to controls that do not have another DataSource assigned (i.e. for controls having DataSource = *nil*). Otherwise, it will be assigned to all data-aware controls in Editor 686.

6.2.3 TRVOfficeConverter

TRVOfficeConverter allows using text converters from Microsoft Office (32-bit versions) with TRichViews, for importing and exporting files in different formats.

The converters technology is **obsolete**: new versions of Microsoft Office do not use them. But old converters still may be used.

This component is not available in FireMonkey version.

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 691 and ExportConverters 690 properties.

If there are no converters installed, converters lists are empty.

GetImportFilter 693 and GetExportFilter 693 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 691 to file in another format (ExportRTF 692), or you can convert file to RTF Stream (ImportRTF 694).
2. You can export content of TRichView component in a file (ExportRV 693) or import from file (ImportRV 694). Such conversion still uses Stream 691 internally.

While converting, the component generates OnConverting 695 event.

If PreviewMode 691 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*\OfficeConverters\

Additional resources

- A set of converters for new formats of Microsoft Office: ***download from our web site*** (this file was originally placed on the Microsoft's website).

6.2.3.1 Properties

In TRVOfficeConverter

- ErrorCode 689
- ExcludeDocXExportConverter 690
- ExcludeDocXImportConverter 690
- ExcludeHTMLExportConverter 690
- ExcludeHTMLImportConverter 690
- ExportConverters 690
- ExtensionsInFilter 690
- ImportConverters 691
- PreviewMode 691
- Stream 691

6.2.3.1.1 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

6.2.3.1.2 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 [691].
If ExcludeDocXExportConverter=*True*, DocX export converter is excluded from the list of ExportConverters [690] (TRichView has SaveDocX [341] method allowing to save DocX better than this converter).

Default value

False

6.2.3.1.3 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 [691] (use TRichView.LoadHTML [330]).
If **ExcludeHTMLExportConverter**=*True*, HTML export converter is excluded from the list of ExportConverters [690] (use TRichView.SaveHTML [342]).

Default value

False

6.2.3.1.4 TRVOfficeConverter.ExportConverters

Collection of installed export converters. Export converters convert RTF to foreign formats.

```
property ExportConverters: TRVOfficeCnvList [696];
```

This is a list of TRVOfficeConverterInfo [696].

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 [691] property;
- GetExportFilter [693] method.

6.2.3.1.5 TRVOfficeConverter.ExtensionsInFilter

Allows including file extensions in the file filter (GetImportFilter [693] and GetExportFilter [693])

```
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

6.2.3.1.6 TRVOfficeConverter.ImportConverters

Collection of installed import converters. Import converters convert foreign formats to RTF.

```
property ImportConverters: TRVOfficeCnvList[696];
```

This is a list of TRVOfficeConverterInfo[696].

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[690] property;
- GetImportFilter[693] method.

6.2.3.1.7 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

6.2.3.1.8 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[692] and ImportRTF[694] work with this stream directly.

The methods ExportRV[693] and ImportRV[694] use this stream internally.

6.2.3.2 Methods

In TRVOfficeConverter

Create[692]
Destroy[692]
ExportRTF[692]
ExportRV[693]
GetExportFilter[693]
GetImportFilter[693]

ImportRTF[694]
ImportRV[694]
IsValidImporter[695]

6.2.3.2.1 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.

6.2.3.2.2 TRVOfficeConverter.Destroy

Destroys the RVOfficeConverter

```
destructor Destroy;
```

6.2.3.2.3 TRVOfficeConverter.ExportRTF

Exports RTF from Stream[691] to file in foreign format.

```
function ExportRTF(const FileName: TRVUnicodeString[1099];
  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[690] list).

When calling this method, Stream[691] must contain RTF (Rich Text Format) document. The method start conversion from Stream[691].Position and stopped when reached the end of stream.

You can provide a visual indication for the conversion progress using OnConverting[695] 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[694];
- ExportRV[693].

See also properties:

- ErrorCode[689].

6.2.3.2.4 TRVOfficeConverter.ExportRV

Exports content of **rv** to file in foreign format.

```
function ExportRV(const FileName: TRVUnicodeString[1099];
  rv: TCustomRichView[216]; 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[690] list)

This method:

1. clears Stream[691];
2. saves RTF to Stream[691] using rv.SaveRTFToStream[351];
3. performs converting;
4. clears Stream[691].

You can provide a visual indication for the conversion progress using OnConverting[695] 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[694];
- ExportRV[693].

See also properties:

- ErrorCode[689].

6.2.3.2.5 TRVOfficeConverter.GetExportFilter

Returns string that can be assigned to the Filter property of TSaveDialog.

```
function GetExportFilter: TRVUnicodeString[1099];
```

(changed in version 18)

If ExtensionsInFilter[690] is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- GetImportFilter[693] method;
- ExportConverters[690] property.

6.2.3.2.6 TRVOfficeConverter.GetImportFilter

Returns string that can be assigned to the Filter property of TOpenDialog

```
function GetImportFilter: TRVUnicodeString[1099];
```

(changed in version 18)

If ExtensionsInFilter[690] is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- GetExportFilter[693] method;
- ImportConverters[691] property.

6.2.3.2.7 TRVOfficeConverter.ImportRTF

Converts data from the file in foreign format to RTF data in Stream[691]

```
function ImportRTF(const FileName: TRVUnicodeString[1099];
  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[691] list)

The method clears Stream[691] and writes RTF (Rich Text Format) document in it.

You can provide a visual indication for the conversion progress using OnConverting[695] 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[692];
- ImportRV[694].

See also properties:

- ErrorCode[689].

6.2.3.2.8 TRVOfficeConverter.ImportRV

Imports content of file in foreign format to **rv**.

```
function ImportRV(const FileName: TRVUnicodeString[1099];
  rv: TCustomRichView[216]; 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[691] list)

This method:

1. clears Stream[691];
2. imports data from the file to Stream[691];
3. reads data from Stream[691] in rv, using rv.LoadRTFFromStream[335],

4. clears Stream[691].

You can provide a visual indication for the conversion progress using OnConverting[695] 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[694];
- ExportRV[693].

See also properties:

- ErrorCode[689].

6.2.3.2.9 TRVOfficeConverter.IsValidImporter

Checks if the specified file can be imported with the specified converter.

```
function IsValidImporter(const FileName: TRVUnicodeString[1099];
  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[691] 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[689].

6.2.3.3 Events

In TRVOfficeConverter

- OnConverting[695]

6.2.3.3.1 TRVOfficeConverter.OnConverting

Occurs while converting (in ImportRTF[694], ExportRTF[692], ImportRV,[694] ExportRV[693] 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 [392].

6.2.3.4 Classes of properties

Classes of TRVOfficeConverter [687] Properties

- TRVOfficeCnvList [696] – a list of TRVOfficeConverterInfo [696] items representing properties of import or export converters. This is a type of ImportConverters [691] and ExportConverters [690] properties.

6.2.3.4.1 TRVOfficeCnvList

This class represents a list of installed import [691] or export [690] 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 [696]).

6.2.3.4.2 TRVOfficeConverterInfo

This class represents an item in list of installed import [691] and export [690] 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').

6.2.4 TRVPrint

This component allows printing TRichView [216], TRichViewEdit [469], TDBRichView [604], or TDBRichViewEdit [616]. It is Invisible at run-time.

Unit [VCL/FMX] PtblRV / fmxPtblRV.

Syntax

```
TRVPrint = class(TCustomRVPrint 877)
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVPrint 877

How to Use

Printing or drawing pages

1. Assign the source component for printing (AssignSource 709 method).
2. (optionally) Assign headers and/or footers (SetHeader 716, SetFooter 715).
3. Format pages (FormatPages 711 method).
4. Now you can:

- get a number of pages (PagesCount 882 property);
- display a print preview (using TRVPrintPreview 635 component; see also DrawPreview 711, MakePreview 712, MakeScaledPreview 713, DrawPage, DrawPageAt 710 methods);
- print the whole document (Print 713 method) or range of pages (PrintPages 714 method).

5. When finished, you can release some temporally allocated memory (Clear 883).

PDF saving

1. Check of PDF saving is available (CanSavePDF 883).
2. Assign the source component for printing (AssignSource 709 method).
3. (optionally) Assign headers and/or footers (SetHeader 716, SetFooter 715).
4. Save PDF (SavePDF 715)
5. When finished, you can release some temporally allocated memory (Clear 883).

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 711.

Alternatively, you can use "virtual printer" to draw pages on any canvas. Assign VirtualPrinter 707 .Active 721 = *True* to use parameters (page size and DPI) specified in VirtualPrinter 707, instead of parameters of the current printer.

Events

You can display some information about the process of paginating and printing:

- OnFormatting 717 occurs during paginating (FormatPages 711 method);
- OnSendingToPrinter 719 occurs during printing (Print 713 and PrintPages 714 methods).

You can draw additional content using the following events:

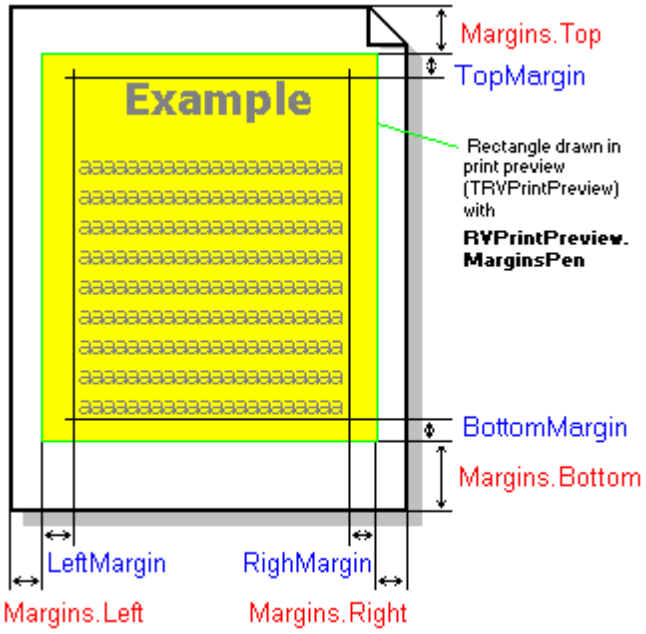
- OnPagePrepaint[719] and OnPagePostpaint[718] allow drawing on each page below/over TRichView document;
- OnPrintComponent[887] requests an image of an inserted control for printing.

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[885]
- OnDrawHyperlink[886]

Margins

Margins are specified in Margins[705] property.



Margins in TRVPrint

This picture illustrates the meaning of margin properties. Paragraph indents are not shown for simplification.

Margins[705] are measured in Units[707] (shown in red).

TRichView.LeftMargin[245], TRichView.RightMargin[251], TRichView.TopMargin[262] and TRichView.BottomMargin[231] are measured in “standard” pixels (1 pixel = 1/Style[260].UnitsPixelsPerInch[765] of an inch; shown in blue).

Left and right margins can be swapped for even pages, if MirrorMargins[706] is set to *True*.

See also ClipMargins[703] property.

The picture above is for VCL and LCL version of the components.

FireMonkey version uses MarginsStroke[892] property instead of MarginsPen property.

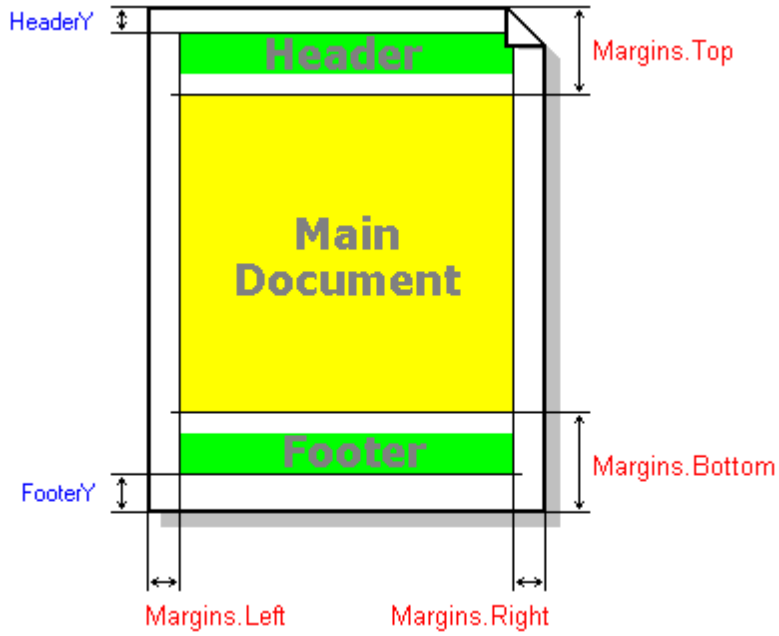
Headers and Footers

TRVPrint supports 3 types of headers and footers:

- normal header and footer
- header and footer for the first page (used only if TitlePage [706]=*True*)
- header and footer for even pages (used only if FacingPages [703]=*True*)

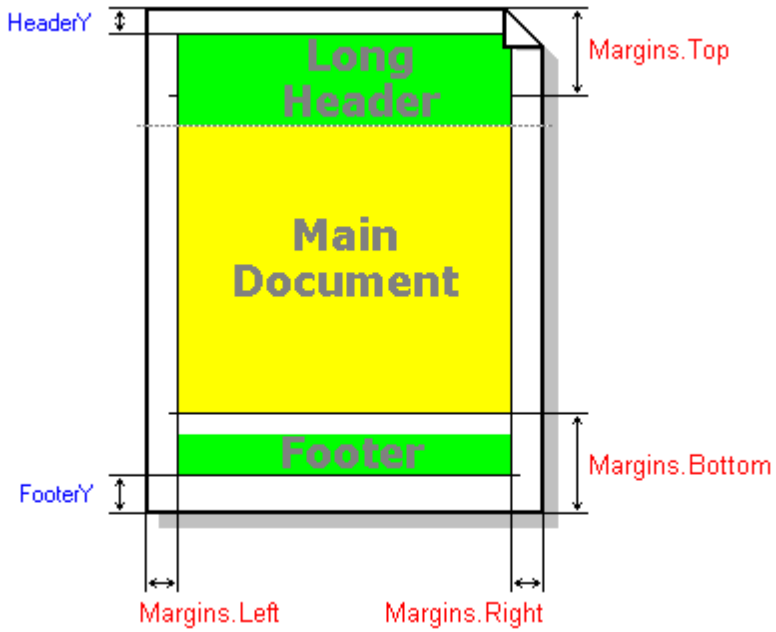
Headers are assigned using SetHeader [716], footers are assigned using SetFooter [715] methods.

Any TRichView document can be assigned as a header or footer. Positions of headers and footers are defined by HeaderY [705] and FooterY [704].



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[719] event;
- using TRVReportHelper[721].

See also: GetHeaderRect[712], GetFooterRect[712].

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[880];
- NoMetafiles[881].

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[233],
- BackgroundProperties[229],
- BackgroundPicture[229].

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[881] and MaxPrintedItemNo[879] 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[719] event
3. Header's text boxes[188] and sidenotes[186] having PositionInText[1001]=*rvpitBelowText*
4. Header[716]
5. Header's text boxes and sidenotes having PositionInText=*rvpitAboveText*
6. Footer's text boxes and sidenotes having PositionInText=*rvpitBelowText*
7. Footer[716]
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[718] event

Text boxes and sidenotes are drawn in the order of their positions in the parent document.

Properties specific for FireMonkey version

The following properties are used only in FireMonkey:

- BestDPI[702].

Tips and Tricks

See ***https://www.trichview.com/forums/viewtopic.php?p=250***.

See Also

See also:

- UpdatePaletteInfo[884] method;
- TRichView.PageBreaksBeforeItems[250];
- TRichView.AssignSoftPageBreaks[284];
- TRVReportHelper[721] components;
- RichViewAlternativePicPrint[1110] typed constant;
- RichViewPixelsPerInch[1115] typed constant.

Demo projects:

- Demos*\Assorted\Printing\

6.2.4.1 Properties

In TRVPrint

- BestDPI 702 [FMX]
- ClipMargins 703
- FixMarginsMode 703
- FooterY 704
- HeaderY 705
- Margins 705
- MirrorMargins 706
- Preview100PercentHeight 706
- Preview100PercentWidth 706
- TitlePage 706
- Units 707
- VirtualPrinter 707

Derived from TCustomRVPrint 877

- ColorMode 878
- DarkMode 878
- EndAt 878
- FirstPageNo 879
- IgnorePageBreaks 879
- LastPageNo 879
- MaxPrintedItemNo 879
- MetafileCompatibility 880
- MetafilePixelsPerInch 880
- MinPrintedItemNo 881
- NoMetafiles 881
- PageNoFromNumber 881
- PagesCount 882
- PreviewCorrection 882
- TransparentBackground 882

6.2.4.1.1 TRVPrint.BestDPI

Specifies the preferred printer DPI ("dots per inch") value.

```
property BestDPI: TRVPrinterDPI;
```

(introduced in version 20)

This property is used if the printer's DPI is not specified (i.e. if Printer.ActivePrinter.ActiveDPIIndex < 0).

Selects the DPI that best fits **BestDPI** and activates it.

BestDPI has two integer fields: X and Y. They define horizontal and vertical DPI values. If one of them = 0, the component selects the first DPI listed in Printer.ActivePrinter.DPI[].

If Printer.ActivePrinter.ActiveDPIIndex is already specified, this property is ignored.

Default value:
600, 600

6.2.4.1.2 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

6.2.4.1.3 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[706];
- MirrorMargins[706].

See also methods:
- SetHeader[716];
- SetFooter[715];
- AssignDocParameters[708].

6.2.4.1.4 TRVPrint.FixMarginsMode

Specifies how to work with too small Margins[705] (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

6.2.4.1.5 TRVPrint.FooterY

Vertical position of a page footer.

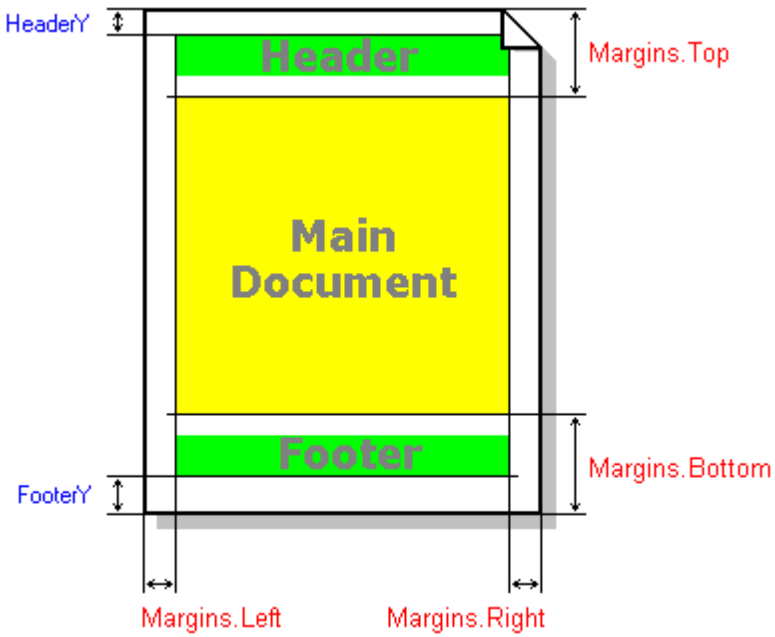
```
property FooterY: TRVLength[1081];
```

(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[707].

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[705].

See also methods:

- SetFooter[715];
- AssignDocParameters[708].

6.2.4.1.6 TRVPrint.HeaderY

Vertical position of a page header.

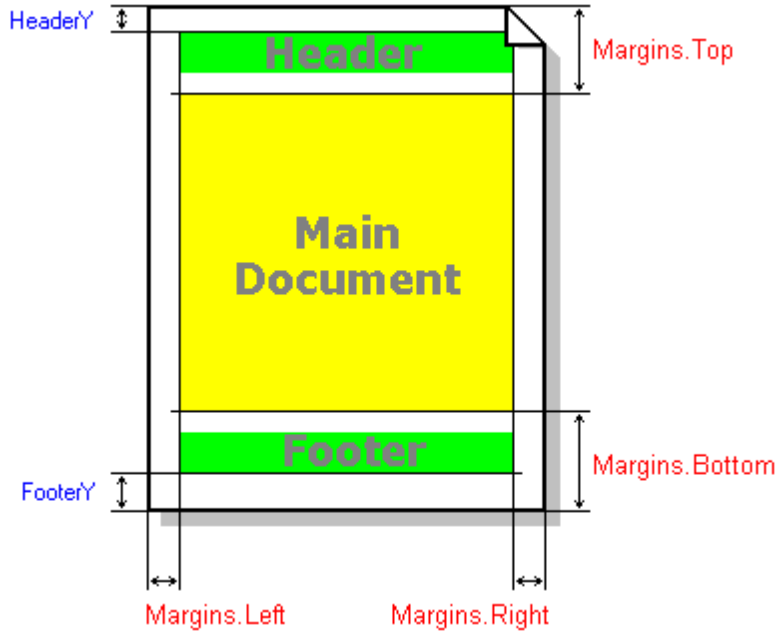
```
property HeaderY: TRVLength[1081];
```

(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[707].

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[704].

See also methods:

- SetHeader[716];
- AssignDocParameters[708].

6.2.4.1.7 TRVPrint.Margins

Page margins.

```
property Margins: TRVUnitsRect[1042];
```

(introduced in version 15)

Margins are measured in Units[707].

If MirrorMargins[706]=*True*, Margins.Left and Margins.Right are swapped for even pages.

You need to reformat pages after changing values of this property, see FormatPages[711].

See also the picture in the TRVPrint[696] main topic.

Default value:

(20.0, 20.0, 20.0, 20.0)

See also properties:

- HeaderY[705];
- FooterY[704];

See also methods:

- AssignDocParameters[708].

6.2.4.1.8 TRVPrint.MirrorMargins

If set to *True*, Margins[705].Left and Margins[705].Right are swapped for even pages.

```
property MirrorMargins: Boolean;
```

(introduced in version 1.6)

Default value:

False

See also properties:

- FacingPages[703].

See also methods:

- AssignDocParameters[708].

6.2.4.1.9 TRVPrint.Preview100PercentHeight

Height of a page when drawing a preview on the screen at 100% scale, pixels.

```
property Preview100PercentHeight: TRVCoord[1064];
```

Mainly for internal use.

6.2.4.1.10 TRVPrint.Preview100PercentWidth

Width of a page when drawing a preview on the screen at 100% scale, pixels.

```
property Preview100PercentWidth: TRVCoord[1064];
```

Mainly for internal use.

6.2.4.1.11 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 [703].

See also methods:

- SetHeader [716];
- SetFooter [715];
- AssignDocParameters [708].

6.2.4.1.12 TRVPrint.Units

Defines measurement units for sizes in TRVPrint.

```
property Units: TRVUnits[1099];
```

(introduced in version 15)

The following properties are measured in Units:

- HeaderY [705];
- FooterY [704];
- Margins [705].

A direct assignment to this property does not change values of these properties. To convert properties, use ConvertToUnits [710] method.

If Units [707] = *rvuPixels*, one unit = 1/RichView.Style [260].UnitsPerInch [765] of an inch, where RichView is a control assigned for printing by calling AssignSource [709] method. When measuring on paper, they are recalculated according to the printer DPI.

Default value:

rvuMillimeters

6.2.4.1.13 TRVPrint.VirtualPrinter

Defines properties of a "virtual printer"

```
property VirtualPrinter: TRVVirtualPrinterProperties[720];
```

(introduced in version 17)

If **VirtualPrinter**.Active [721] = *True*, TRVPrint prepares pages for drawing on any canvas, using page size [721] and DPI value [721] 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 [710] methods. Additionally, if PDF saving engine is available, you can save document as a PDF file [715].

"Virtual printer" functionality is similar to the functionality provided by TRVReportHelper [721] component. Differences:

- a report helper draws document directly at the specified location; a virtual printer draws a page having the specified margins;
- a report helper does not support headers [716] and footers [715];
- a report helper does not support printing text boxes [188] and side notes [186].

6.2.4.2 Methods

In TRVPrint

AssignDocParameters 708
AssignSource 709
ContinuousPrint 709
ConvertToUnits 710
Create 710
DrawPageAt 710
DrawPage 710
DrawPreview 711
FormatPages 711
GetFooterRect 712
GetHeaderRect 712
MakePreview 712
MakeScaledPreview 713
Print 713
PrintPages 714
SavePDF 715
SetFooter 715
SetHeader 716

Derived from TCustomRVPrint 877

CanSavePDF 883
Clear 883
GetFirstItemOnPage 883
GetPageNo 883
UpdatePaletteInfo 884

6.2.4.2.1 TRVPrint.AssignDocParameters

Assigns properties from **DocParameters**.

```
procedure AssignDocParameters(DocParameters: TRVDocParameters 422);
```

(introduced in version 10)

This procedure assigns the following properties:

- Margins 705;
- HeaderY 705, FooterY 704;
- MirrorMargins 706;
- FacingPages 703;
- TitlePage 706.

Sizes are converted from **DocParameters**.Units 428 to Units 707 (the value of Units 707 property itself is not changed). When converting from/to pixels, this method uses DPI specified in RichView.Style 260.UnitsPerInch 765, where RichView is a control assigned for printing by calling AssignSource 709 method (or 96, if AssignSource was not called yet).

This method does not assign printer properties (page size and orientation).

6.2.4.2.2 TRVPrint.AssignSource

Assigns document for printing.

```
procedure AssignSource(PrintMe: TCustomRichView[216]);
```

You can print document in the following components:

- TRichView[216],
- TRichViewEdit[469],
- TDBRichView[604],
- TDBRichViewEdit[616].

Document is not copied to RVPrint, so do not modify and do not destroy it until printing is completed.

See also:

- SetHeader[716];
- SetFooter[715].

6.2.4.2.3 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[713] or PrintPages[714] methods.

Do not call this method if VirtualPrinter[707].Active[721] = *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[882] := 0;
RVPrint.TransparentBackground[882] := True;
RVPrint.AssignSource[709](RichView1);
RVPrint.FormatPages[711](rvdoALL);
RVPrint.ContinuousPrint;
// printing RichView2
RVPrint.StartAt[882] := RVPrint.EndAt[878];
RVPrint.AssignSource[709](RichView2);
RVPrint.FormatPages[711](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[709](RichView1);
RVPrint.FormatPages[711](rvdoALL);
```

```
RVPrint.ContinuousPrint;
// printing RichView2 in landscape orientation
Printer.NewPage;
Printer.Orientation := poLandscape;
RVPrint.AssignSource(RichView2);
RVPrint.FormatPages(rvdoALL);
RVPrint.ContinuousPrint;
Printer.EndDoc;
```

6.2.4.2.4 TRVPrint.ConvertToUnits

Converts properties to new units.

```
procedure ConvertToUnits(AUnits: TRVUnits);
```

(introduced in version 15)

The method converts HeaderY, FooterY, Margins from Units to **AUnits**, then assigns **AUnits** to Units.

6.2.4.2.5 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.

6.2.4.2.6 TRVPrint.DrawPage, DrawPageAt

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.

You need to assign source document for printing (AssignSource method) and format pages (FormatPages method) before calling this method.

See also methods:

- DrawPreview.

See also properties:

- PreviewCorrection;
- PagesCount.

See also events:

- OnPrintComponent;

- OnPagePrepaint[719];
- OnPagePostpaint[718].

6.2.4.2.7 TRVPrint.DrawPreview

Draws print preview of the **pgNo**-th page to **PageRect** rectangle of **Canvas**.

```
procedure DrawPreview(pgNo: Integer; Canvas:  TCanvas;
  const PageRect: TRVCoordRect[1064]);
```

There is more convenient way to display print preview: TRVPrintPreview[635] component.

The page image is scaled to fit **PageRect**. To maintain correct proportions, you can use Preview100PercentWidth[706] and Preview100PercentHeight[706] properties.

pgNo must be in range from 1 to PagesCount[882].

You need to assign source document for printing (AssignSource[709] method) and format pages (FormatPages[711] method) before calling this method.

To draw unscaled page, use DrawPage and DrawPageAt[710] methods.

See also methods:

- MakePreview[712];
- MakeScaledPreview[713];
- DrawPage, DrawPageAt[710].

See also properties:

- PreviewCorrection[882];
- PagesCount[882].

See also events:

- OnPrintComponent[887];
- OnPagePrepaint[719];
- OnPagePostpaint[718].

6.2.4.2.8 TRVPrint.FormatPages

Prepares a document for printing (or drawing)

```
function FormatPages(PrintOptions:TRVDisplayOptions[1065]): Integer;
```

Please use *rvdoAll* constant for **PrintOptions** parameter:

```
FormatPages(rvdoAll);
```

Document must be assigned to this TRVPrint object using AssignSource[709] method before calling this method.

This method allocates some temporal memory what is used when drawing preview and printing. When you finish, call Clear[883] to release this memory.

Note for C++Builder users:

Older versions of C++Builder translates rvdoAll constant incorrectly.

The recommended call is

```
RVPrint->FormatPages(TRVDisplayOptions())
```

(i.e. initialization with the empty set; this parameter is for compatibility, and it is not used anymore)

See also methods:

- Clear[883].

See also events:

- OnFormatting[717].

6.2.4.2.9 TRVPrint.GetFooterRect

Returns coordinates of the footer for the page **PageNo** (from 1), in printer pixels

```
function GetFooterRect(PageNo: Integer): TRVCoordRect[1064];
```

(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[711]).

This function can be used in OnPagePrepaint[719] and OnPagePostpaint[718] events.

See also:

- SetFooter[715];
- GetHeaderRect[712].

6.2.4.2.10 TRVPrint.GetHeaderRect

Returns coordinates of the header for the page **PageNo** (from 1), in printer pixels

```
function GetHeaderRect(PageNo: Integer): TRVCoordRect[1064];
```

(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[711])

This function can be used in OnPagePrepaint[719] and OnPagePostpaint[718] events.

See also:

- SetHeader[716];
- GetFooterRect[712].

6.2.4.2.11 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[635] component.

pgNo must be in range from 1 to PagesCount[882].

This method changes size of bitmap to (Preview100PercentWidth[706] x Preview100PercentHeight[706]) and draws print preview of the **pgNo**-th page into it.

You need to assign source document for printing (AssignSource[709] method) and format pages (FormatPages[711] method) before calling this method.

See also methods:

- MakeScaledPreview[713];

- DrawPreview[711].

See also properties:

- PreviewCorrection[882];
- PagesCount[882].

See also events:

- OnPrintComponent[887];
- OnPagePrepaint[719];
- OnPagePostpaint[718].

6.2.4.2.12 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[635] component.

pgNo must be in range from 1 to PagesCount[882].

This method does not change size of bitmap, and draws the page in full bitmap size. To maintain correct proportions, you can use Preview100PercentWidth[706] and Preview100PercentHeight[706] properties.

You need to assign source document for printing (AssignSource[709] method) and format pages (FormatPages[711] method) before calling this method.

See also methods:

- MakePreview[712];
- DrawPreview[711].

See also properties:

- PreviewCorrection[882];
- PagesCount[882].

See also events:

- OnPrintComponent[887];
- OnPagePrepaint[719];
- OnPagePostpaint[718].

6.2.4.2.13 TRVPrint.Print

Prints the whole document

```
procedure Print(Title: String; Copies: Integer; Collate: Boolean;
  PageSet: TRVPageSet[1083] = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo[881] and MaxPrintedItemNo[879] 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 [709] method) and format pages (FormatPages [711] method) before calling this method.

```
Print(Title, Copies, Collate, PageSet, Ascending)
```

is equivalent to

```
PrintPages[714](1, PagesCount[882], Title, Copies, Collate, PageSet, Ascending)
```

Do not call this method if VirtualPrinter [707].Active [721] = *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 [882] is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- PrintPages [714];

See also events:

- OnSendingToPrinter [719];
- OnPrintComponent [887];
- OnPagePrepaint [719];
- OnPagePostpaint [718].

6.2.4.2.14 TRVPrint.PrintPages

Prints the range of pages, from **FirstPgNo** to **LastPgNo**

```
procedure PrintPages(FirstPgNo, LastPgNo: Integer; Title: String;
  Copies: Integer; Collate: Boolean;
  PageSet: TRVPageSet[1083] = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo [881] and MaxPrintedItemNo [879] properties into account.

Parameters

FirstPgNo – the first page to print, must be in range 1..PagesCount [882].

LastPgNo – the last page to print, must be in range 1..PagesCount [882].

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[709] method) and format pages (FormatPages[711] method) before calling this method.

Do not call this method if VirtualPrinter[707].Active[721] = *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[882] is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- Print[713].

See also events:

- OnSendingToPrinter[719];
- OnPrintComponent[887];
- OnPagePrepaint[719];
- OnPagePostpaint[718].

6.2.4.2.15 TRVPrint.SavePDF

Saves a document as a PDF file.

```
function SavePDF(const FileName: TRVUnicodeString[1099];
  PDFInfo: PRVPDFMetadata[1084] = nil): Boolean;
```

(introduced in v22)

This method works only if a PDF saving engine is available. Currently, it is implemented only in FireMonkey version, using Skia4Delphi[163]. You can check availability of a PDF saving engine using CanSavePDF[883] method.

This method requires an active virtual printer mode[707]. It uses VirtualPrinter[707].PageWidth and PageHeight[721] properties.

Internally, this method calls FormatPages[711] method.

Parameters

FileName – output file name

PDFInfo – pointer to TRVPDFMetadata[1084] structure containing PDF creation parameter. This parameter is optional, use nil to create PDF with default parameters.

See:

- TRVReportHelper[721].SavePDF[727].

6.2.4.2.16 TRVPrint.SetFooter

Assigns a page footer.

```
procedure SetFooter(RVData: TCustomRVData[1014];
  FooterType: TRVHFType[1078]=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 [711].

Using different footers

A footer for the first page is used only if TitlePage [706]=*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 [703]=*True*, otherwise it is ignored, and a normal footer is used on both odd and even pages.

Example:

```
var
  rvMain, rvFooter: TRichView[216];
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetFooter(rvFooter.RVData[254]);
RVPrint1.TitlePage[706] := True;
// hiding footer on the first page
RVPrint1.SetFooter(nil, rvhftFirstPage);
```

See also:

- SetHeader [716];
- AssignSource [709].

See also properties:

- FooterY [704].

See also:

- TRichView [216].SetFooter [362].

6.2.4.2.17 TRVPrint.SetHeader

Assigns a page header.

```
procedure SetHeader(RVData: TCustomRVData[1014];
  FooterType: TRVHFType[1078]=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 [711].

Using different headers

A header for the first page is used only if TitlePage[706]=*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[703]=*True*, otherwise it is ignored, and a normal header is used on both odd and even pages.

Example:

```
var
  rvMain, rvHeader: TRichView[216];
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetHeader(rvHeader.RVData[254]);
RVPrint1.TitlePage[706] := True;
// hiding header on the first page
RVPrint1.SetHeader(nil, rvhftFirstPage);
```

See also:

- SetFooter[715];
- AssignSource[709].

See also properties:

- HeaderY[705].

See also:

- TRichView[216].SetHeader[363].

6.2.4.3 Events

In TRVPrint

- OnFormatting[717]
- OnPagePostpaint[718]
- OnPagePrepaint[719]
- OnSendingToPrinter[719]

Derived from TCustomRVPrint[877]

- OnAfterPrintImage[885]
- OnDrawCheckpoint[885]
- OnDrawHyperlink[886]
- OnPrintComponent[887]

6.2.4.3.1 TRVPrint.OnFormatting

Occurs while FormatPages[711] method is executed.

```
type
  TRVPrintingEvent = procedure(Sender: TCustomRichView[216];
    PageCompleted: Integer;
    Step:TRVPrintingStep[1085]) 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 [711].

See also events:

- OnSendingToPrinter [719].

6.2.4.3.2 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: TRVCoordRect[1064]) of object;

property OnPagePostpaint: TRVPagePrepaintEvent;
```

(introduced in version 1.7)

Sequence of actions:

1. drawing background (if not transparent);
2. OnPagePrepaint [719];
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 [712];
- GetFooterRect [712];

- the picture in the TRVPrint[696] main topic.

6.2.4.3.3 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: TRVCoordRect[1064]) 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[718].

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[712];
- GetFooterRect[712];
- the picture in the TRVPrint[696] main topic;
- example how to print pictures in this event: ***https://www.trichview.com/forums/viewtopic.php?t=58***.

6.2.4.3.4 TRVPrint.OnSendingToPrinter

Occurs while methods Print[713] or PrintPages[714] are executed.

```
type
  TRVPrintingEvent = procedure(Sender: TCustomRichView[216];
    PageCompleted: Integer; Step:TRVPrintingStep[1085]) 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[713];
- PrintPages[714].

See also events:

- OnFormatting[717].

6.2.4.4 Classes of properties

Classes of TRVPrint[696] Properties

- TRVVirtualPrinterProperties[720] is a type of TRVPrint.VirtualPrinter[707] property. It contains properties controlling printing on the specified canvas.

6.2.4.4.1 TRVVirtualPrinterProperties

This is a class of VirtualPrinter[707] property of TRVPrint[696].

Unit RVStyle.

Syntax

```
TRVVirtualPrinterProperties = class(TPersistent)
```

Hierarchy

TObject
TPersistent

Main Properties

- Active[721] enables/disables a virtual printer
- PageWidth, PageHeight[721]
- PixelsPerInch[721]

6.2.4.4.1.1 Properties

In TRVVirtualPrinterProperties

- Active[721]
- PageHeight[721]
- PageWidth[721]

- PixelsPerInch [721]

Enables the virtual printer.

```
property Active: Boolean;
```

TRVPrint [696] 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 pixels in a virtual inch (DPI)

```
property PixelsPerInch: Boolean;
```

If this value is positive, it is used as a DPI of the virtual printer.

Otherwise, if RichViewPixelsPerInch [1115] 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 at 96 dpi)

6.2.5 TRVReportHelper

This component is similar to TRVPrint [696], 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 [707];
- it contains a document, not a reference to it.

Unit [VCL/FMX] RVReport / fmxRVReport.

Syntax

```
TRVReportHelper = class(TCustomRVPrint [877])
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVPrint [877]

How to Use

1. Load document in RVReportHelper.RichView[724].
2. Initialize formatting (Init[727]). Specify the page width and canvas.
3. Format the document page by page (FormatNextPage[726]). Pages may have different heights.
4. Draw pages on a canvas (DrawPage[725] or DrawPageAt[725]). If you need to know positions of hyperlinks and checkpoints, use OnDrawHyperlink[886] and OnDrawCheckpoint[885] events.

Example (drawing document in a metafile)

```
procedure TForm1.DrawToMetafile(rvh: TRVReportHelper;
  wmf: TMetafile; Width: Integer);
const VERYLARGEVALUE = $FFFFFFF;
var Canvas: TMetafileCanvas;
begin
  rvh.IgnorePageBreaks[879] := True;
  rvh.Init[727](Self.Canvas, Width);
  while rvh.FormatNextPage[726](VERYLARGEVALUE) do;
  wmf.Width := Width;
  wmf.Height := rvh.GetLastPageHeight[726];
  Canvas := TMetafileCanvas.Create(wmf, 0);
  rvh.DrawPage[725](1,Canvas,True,rvh.GetLastPageHeight[726]);
  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[880];
- NoMetafiles[881].

FireMonkey notes

FireMonkey version

Headers and footers

TRVReportHelper does not support headers and footers.

If you want to store them, you can use TRVReportHelperWithHeaderFooters[1038] class (or use ScaleRichView).

Demo projects

- Demos*\Assorted\Graphics\ToImage\
- Demos*\Assorted\Printing\ReportHelper\

6.2.5.1 Properties

In TRVReportHelper

- AllowTransparentDrawing[723]
- RichView[724]

- TargetPixelsPerInch [723] [VCL,LCL]
- TargetPixelsPerInchX [723] [FMX]
- TargetPixelsPerInchY [723] [FMX]

Derived from TCustomRVPrint [877]

- ColorMode [878]
- DarkMode [878]
- EndAt [878]
- FirstPageNo [879]
- IgnorePageBreaks [879]
- LastPageNo [879]
- MaxPrintedItemNo [879]
- MetafileCompatibility [880]
- MetafilePixelsPerInch [880]
- MinPrintedItemNo [881]
- NoMetafiles [881]
- PageNoFromNumber [881]
- PagesCount [882]
- PreviewCorrection [882]
- TransparentBackground [882]

6.2.5.1.1 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 [725] /DrawPageAt [725] 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

6.2.5.1.2 TRVReportHelper.TargetPixelsPerInch

Overrides the scaling factor for drawing.

VCL and LCL:

```
property TargetPixelsPerInch: Integer;
```

(introduced in version 19)

Assign a positive value to override a scaling factor ("pixels per inch" value) for the target canvas.

If **TargetPixelsPerInch** = 0, the component draws using PPI of the specified Canvas.

If **TargetPixelsPerInch** > 0, value of this property is used as PPI. As a side affect, the component assigns TargetPixelsPerInch to Canvas.Font.PixelsPerInch for canvas parameters of Init[727], DrawPage[725], DrawPageAt[725] methods.

All sizes are converted from RichView[724].Style[260].UnitsPixelsPerInch[765] to the target canvas PPI.

FireMonkey:

```
property TargetPixelsPerInchX: Integer;
property TargetPixelsPerInchY: Integer;
```

The meaning is the same as for VCL version, but you can assign different values in horizontal (**TargetPixelsPerInchX**) and vertical (**TargetPixelsPerInchY**) directions. These properties are used only if they both have positive values.

In FireMonkey version, these properties are very important if you use TRVReportHelper to draw on a printer's Canvas. Assign Printer.ActivePrinter.ActiveDPI to these properties before calling Init[727].

Default value

0 (using the Canvas pixel depth)

6.2.5.1.3 TRVReportHelper.RichView

This is the object containing document of the report helper.

```
property RichView: TRichView[216];
```

This RichView object is invisible. Use it to load document and to set properties for loading. This RichView must be linked[260] with some TRVStyle[739] 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[727] and FormatNextPage[726] instead.

6.2.5.2 Methods

In TRVReportHelper

DrawPage[725]
DrawPageAt[725]
Finished[726]
FormatNextPage[726]
GetLastPageHeight[726]
Init[727]
SavePDF[727]
UpdatePageNumbers[727]

Derived from TCustomRVPrint [877]

Clear [883]
GetFirstItemOnPage [883]
GetPageNo [883]
UpdatePaletteInfo [884]

6.2.5.2.1 TRVReportHelper.DrawPage

Draws the specified page

```
procedure DrawPage(APageNo: Integer; ACanvas: TCanvas;
  APreview: Boolean; AHeight: TRVCoord[1064]);
```

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 [727], 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 [726].
If value of TargetPixelsPerInch [723] is positive, it is assigned to ACanvas.Font.PixelsPerInch.

See also:

- DrawPageAt [725].

6.2.5.2.2 TRVReportHelper.DrawPageAt

Draws the specified page at the specified coordinates

```
procedure DrawPageAt(Left, Top: TRVCoord[1064]; APageNo: Integer;
  ACanvas: TCanvas; APreview: Boolean; AHeight: TRVCoord[1064];
  RelativeToCurrentOrigin: Boolean = False);
```

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 [727], 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. If **RelativeToCurrentOrigin** = *False*, the corner is exactly at (**Left**,**Top**). If **RelativeToCurrentOrigin** = *True*, the corner is shifted by (**Left**,**Top**) from the current origin (that was set by the previous call of GetWindowOrgEx).

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[726].
If value of TargetPixelsPerInch[723] is positive, it is assigned to ACanvas.Font.PixelsPerInch.

See also:

- DrawPage[725].

6.2.5.2.3 TRVReportHelper.Finished

Returns *True* if all pages were formatted.

```
function Finished: Boolean;
```

You can use the value returned by FormatNextPage[726] instead.

6.2.5.2.4 TRVReportHelper.FormatNextPage

Formats next page(s) of document

```
function FormatNextPage(AMaxHeight: TRVCoord[1064]): Boolean;
```

Call this method after Init[727] several times, while it returns *True*.

Parameter:

AMaxHeight – page height, pixels; page width is specified as a parameter of Init[727].

When the page is formatted, it can be drawn (DrawPage[725], DrawPageAt[725]).

Sometimes, this method formats more than one page.

This method is fast. The most time consuming method is Init[727].

Return value:

- *True* if not all pages were formatted;
- *False*, if formatting is complete.

6.2.5.2.5 TRVReportHelper.GetLastPageHeight

Returns a real height of the last page, in device pixels

```
function GetLastPageHeight: TRVCoord[1064];
```

This function returns height of document on the last page, when the document is completely formatted (all calls to FormatNextPage[726] are made).

The returned value is measured in **ACanvas** pixels, where **ACanvas** is the parameter of Init[727].

This method is useful if you draw/print several documents one after another.

6.2.5.2.6 TRVReportHelper.Init

Prepares document for formatting.

```
procedure Init(ACanvas: TCanvas; APageWidth: TRVCoord[1064]);
```

Parameters:

ACanvas – canvas object used for formatting.

APageWidth – page width, pixels; all pages have the same width.

VCL and LCL: If value of TargetPixelsPerInch[723] is positive, it is assigned to ACanvas.Font.PixelsPerInch.
FMX: If **ACanvas** is a printer canvas, assign the active printer DPI to (TargetPixelsPerInchX, TargetPixelsPerInchY)[723] properties before calling Init.

See also:

- FormatNextPage[726].

6.2.5.2.7 TRVReportHelper.SavePDF

Saves a document in RichView[724] as a PDF file.

```
function SavePDF(const FileName: TRVUnicodeString;
  PageWidth, PageHeight: Integer;
  PDFInfo: PRVPDFMetadata = nil): Boolean;
```

(introduced in v22)

This method works only if a PDF saving engine is available. Currently, it is implemented only in FireMonkey version, using Skia4Delphi[163]. You can check availability of a PDF saving engine using CanSavePDF[883] method.

Internally, this method calls Init[727] and FormatNextPage[726] methods.

Parameters

FileName – output file name

PageWidth, PageHeight – page size

PDFInfo – pointer to TRVPDFMetadata[1084] structure containing PDF creation parameter. This parameter is optional, use nil to create PDF with default parameters.

See:

- TRVPrint[696].SavePDF[715].

6.2.5.2.8 TRVReportHelper.UpdatePageNumbers

Recalculates page numbers.

```
procedure UpdatePageNumbers;
```

(introduced in version 15)

This method recalculates values of "page number"[190] and "page count"[192] 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[728] event, and call **UpdatePageNumber** before drawing.

6.2.5.3 Events

In TRVReportHelper

- OnGetPageNumber[728]

Derived from TCustomRVPrint[877]

- OnAfterPrintImage[885]
- OnDrawCheckpoint[885]
- OnDrawHyperlink[886]
- OnPrintComponent[887]

6.2.5.3.1 TRVReportHelper.OnGetPageNumber

Request a page number for displaying in "page number" fields[190].

```
type
  TRVGetPageNumberEvent = procedure (Sender: TRVReportHelper[721];
    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[882]), where the "page number" field is located.

DispPageNo – a number that will be displayed in this field. Initially, it is equal to **PageNo**+PageNoFromNumber[881]-1. You can assign another value to this parameter.

See also:

- UpdatePageNumbers[727]

6.2.6 TRVSpellChecker

This component checks spelling in TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[616] controls, as well as in ScaleRichView (TSRichViewEdit, TDBSRichViewEdit controls).

In FireMonkey, this component can also be used as a platform service to check spelling in other controls (like TEdit and TMemo).

Unit [VCL/FMX] RVSpellChecker / fmxRVSpellChecker.

Syntax

```
TRVSpellChecker = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

Availability and Languages

If a spelling checking service is available, IsAvailable[735] returns *True*.

AvailableLanguages[731] returns a list of codes of supported languages (dictionaries). To display them to the user, use FillLanguageNames[734] method.

To select a language (dictionary), assign a language code to Language[731] property, or a language index to LanguageIndex[732] property.

The active language (dictionary) is returned by CurrentLanguage[733] or LangugeIndex[732].

Recommended Use

You can check the editor by registering it using RegisterEditor[736] method.

If you do not use RichViewActions, call Execute[733] method.

If you use RichViewActions:

- place **TRVSpellChecker**, TRVSpellInterface, TRVAControlPanel components on a form;
- assign RVSpellInterface1.SpellChecker = RVSpellChecker1
- assign RVAControlPanel1.SpellInterface = RVSpellInterface1
- add TrvActionSpellingCheck in TActionList and assign it to a button or to a menu item.

The actions add spelling checking command to a popup menu (TRVAPopupMenu), initiates checking on clicking a button or a menu item, localizes spelling checking dialogs.

Alternative Use

You can use “low-level” methods:

- IsWordValid[735] – checks a word for spelling errors (can be used in OnSpellingCheck[416] event of editors)
- Suggest[737] returns suggestions to correct a misspelled word
- AddToAutoReplaceList[733] – “Replace All” command
- AddToIgnoreList[733] – “Ignore All” command
- AddToDictionary[733] – “Add to Dictionary” command

Using as FireMonkey Service

This component can be used as a platform service to check spelling on other FireMonkey editors, see RegisterAsService[735].

Auto-correction

You can use GetAutoCorrection[734] method to get an auto-correction for the specified word.

If you use RichViewActions:

Use TRVSpellInterface component (see “recommended use” above). Call RVSpellInterface1.AutoCorrectInKeyDown in the editor's OnKeyDown event, and RVSpellInterface1.AutoCorrectInKeyPress in the editor's OnKeyPress event.

Word Completion

You can use GuessCompletions [735] method to get possible completions for a word.

This feature is not used by TRichView editors.

Dialog Localization

Unless you use this component via RichViewAction's TRVSpellInterface component, all UI text in spelling checking dialogs are in English.

If you want to localize forms yourself, you can use OnSpellForm [737] event.

Platform Support

Platform	Used Tool
Windows	System spelling checking (ISpellChecker), available since Windows 8
macOS	System spelling checking (NSSpellChecker)
iOS	System spelling checking (UITextChecker)
Linux	Hunspell (may require installation)
Android	–

Not all features available for all platforms, see SupportsFeatures [737].

See Also

Live spelling checking in TRichView [138]

6.2.6.1 Properties

In TRVSpellChecker

- AvailableLanguages [731]
- DialogShown [731]
- Language [731]
- LanguageIndex [732]
- SpellFormStyle [732]

6.2.6.1.1 TRVSpellChecker.AvailableLanguages

Returns a list of available languages.

```
property AvailableLanguages: TStrings;
```

If a spelling checker is not available, this property returns *nil*.

Each item in this property identifiers a language (or a dictionary). A format of items depend on the OS.

To switch the active spelling check language, assign Language [731] or LanguageIndex [732] property.

To get the active spelling check language, use CurrentLanguage [733] method or LanguageIndex [732] property.

To display a list of available languages to the user, use FillLanguageNames [734] method.

6.2.6.1.2 TRVSpellChecker.DialogShown

Returns *True* if a dialog was shown during spelling checking.

```
property DialogShown: Boolean;
```

You can read value of this property after calling Execute [733].

For example, if spelling check was completed without showing a dialog, you can display a message box informing that checking is finished (in RichViewActions, it is implemented in TRVSpellInterface component)

6.2.6.1.3 TRVSpellChecker.Language

Specifies the active spell checking language (dictionary).

```
property Language: String;
```

Assigning value to this property changes the active dictionary.

You can assign:

- one of AvailableLanguages [731]
- an empty string; in this case, the first available language is used.

This property returns the last assigned value.

The actual used language (dictionary) is returned in CurrentLanguage [733] property. If you assigned a correct value to **Language**, CurrentLanguage [733] becomes equal to **Language**.

Platform Notes

In Windows, languages are identified by locale codes according to BCP 47 standard, like 'en-US', 'ru-RU'.

For other OS, underscore characters ('_') are used instead of dashes ('-'), like 'en_US' or 'ru_RU'.

When assigning to Language property, the component automatically adjusts '-' and '_' characters before passing to the spelling checking engine.

Default value

'' (empty string)

See alsoe

6.2.6.1.4 TRVSpellChecker.LanguageIndex

Specifies the index of the active spell checking language (dictionary).

```
property LanguageIndex: Integer;
```

This property is added for a convenience.

A valid range of values for this property is 0 .. AvailableLanguages[731].Count-1.

Assigning to this property changes the active spelling checking language (dictionary). It is identical to assigning Language[731] = AvailableLanguages[731][**LanguageIndex**].

This property returns the index of the currently used spelling checking language (dictionary), i.e. index of CurrentLanguage[733] in AvailableLanguages[731].

6.2.6.1.5 TRVSpellChecker.SpellFormStyle

Defines an appearance of a spelling checking dialog.

```
type
  TRVSpellCheckFormStyle =
    (rvsfsClassic, rvsfsMSWord);

property SpellFormStyle: TRVSpellCheckFormStyle;
```

This dialog is shown by Execute[733] method.

Value	Meaning
rvsfsClassic	In this dialog, the user can modify a misspelled word.
rvsfsMSWord	In this dialog, the user can modify a misspelled word and and text around it.

6.2.6.2 Methods

In TRVSpellChecker

AddToAutoreplaceList[733]
AddToDictionary[733]
AddToIgnoreList[733]
CurrentLanguage[733]
GuessCompletions[735]
Execute[733]
FillLanguageNames[734]
GetAutoCorrection[734]
GetLanguageNameFromCode[735]
IsAvailable[735]
IsWordValid[735]
RegisterAsService[735] [FMX]

RegisterEditor[736]
StartLiveSpelling[736]
Suggest[737]
SupportsFeatures[737]
UnregisterAsService[735] [FMX]
UnregisterEditor[736]

6.2.6.2.1 TRVSpellChecker.AddToAutoreplaceList

Causes occurrences of one word to be replaced by another.

```
procedure AddToAutoreplaceList(const AWordFrom,
  AWordTo: TRVUnicodeString[1099]);
```

This feature is supported not for all platforms. See SupportsFeatures[737].

6.2.6.2.2 TRVSpellChecker.AddToDictionary

Add the specified word to the dictionary.

```
function Suggest(const AWord: TRVUnicodeString[1099]):
  TRVStringList[1041];
```

Treats the provided word as though it were part of the original dictionary.

The word will no longer be considered misspelled, and will also be considered as a candidate for suggestions.

This feature is supported not for all platforms. See SupportsFeatures[737].

6.2.6.2.3 TRVSpellChecker.AddToIgnoreList

Ignores the provided word for the rest of this session.

```
procedure AddToIgnoreList(
  const AWord: TRVUnicodeString[1099]);
```

Note: a list of ignored words is reset when you switch a spelling checking language.

6.2.6.2.4 TRVSpellChecker.CurrentLanguage

Returns the code of the currently used spelling checking language (dictionary)

```
function CurrentLanguage: String;
```

To change the current language (dictionary), assign a new value to Language[731] property. If the assigned language is valid, **CurrentLanguage** will return it.

6.2.6.2.5 TRVSpellChecker.Execute

Starts spelling checking in the specified editor.

```
procedure Execute(Editor: TCustomRichViewEdit[469];
  Scope: TRVEnumScope[1066] = rvesFromCursor);
```

If a potential misspelled word is found, the checker displays a dialog window, where the user can:

- correct this word, or
- correct this word and add it to an auto-correct list, or

- correct this word and add it to a dictionary, or
- ignore this word, or
- ignore all occurrences of this word in this spelling session.

The check is finished when either the end of the document is reached, or the user presses “Cancel” button in the dialog.

A type of dialog window is specified in SpellFormStyle[732] property.

It's recommended to register this Editor with RegisterEditor[736] method, because the spell-checker calls LiveSpellingValidateWord[326] for registered editors when necessary (otherwise, you should process OnSpellFormAction[738] event and call LiveSpellingValidateWord[326] yourself when necessary).

RichViewAction note:

If you use RichViewActions, you do not need to call this method directly. Instead, link this TRVSpellChecker to TRVAControlPanel via TRVSpellInterface component, add TrvActionSpellingCheck to TActionList, link this action to a menu item or a toolbar button.

FireMonkey note:

This method requires Delphi XE7 and newer.

This method can be called in Windows, macOS, Linux (because mobile platforms do not support modal dialogs).

See also:

- DialogShown[731]

6.2.6.2.6 TRVSpellChecker.FillLanguageNames

Add language (dictionary) names to Strings.

```
procedure FillLanguageNames(Strings: TStrings);
```

On return, each string in Strings corresponds to strings of AvailableLanguages[731], but contain not language codes, but language names.

In other words: for each i, Strings.Strings[i] = GetLanguageNameFromCode[735] (AvailableLanguages[731][i]).

6.2.6.2.7 TRVSpellChecker.GetAutoCorrection

Returns an auto-correction for the specified word.

```
function GetAutoCorrection(
  const AWordFrom: TRVUnicodeString[1099];
  out AWordTo: TRVUnicodeString[1099]): Boolean;
```

If the active dictionary has an auto-correction for **AWordFrom**, it is returned in **AWordTo**.

Returns *True* if an auto-correction is found.

This feature is supported not for all platforms. See SupportsFeatures [737].

6.2.6.2.8 TRVSpellChecker.GetLanguageNameFromCode

Returns a language (dictionary) name for displaying to the user.

```
function GetLanguageNameFromCode(
  const Language: String): TRVUnicodeString[1099];
```

Input parameter:

Language is a language code, normally one of AvailableLanguages [731].

Return value:

A corresponding language (or dictionary) name for displaying to the user. If the component cannot retrieve such a name, it returns **Language** parameter.

6.2.6.2.9 TRVSpellChecker.GuessCompletions

Offers possible completions for **AWordStart**.

```
function GuessCompletions(
  const AWordStart: TRVUnicodeString[1099]): TRVStringList[1041];
```

This feature is supported not for all platforms. See SupportsFeatures [737].

6.2.6.2.10 TRVSpellChecker.IsAvailable

Indicates if a spelling checking is available.

```
function IsAvailable: Boolean;
```

Returns *True* is spelling checking can be performed.

Returns *False* otherwise.

6.2.6.2.11 TRVSpellChecker.IsWordValid

Checks the spelling of the specified word.

```
function IsWordValid(
  const AWord: TRVUnicodeString): Boolean;
```

The function returns *True* if:

- this word is valid, or
- spelling checking is not available

6.2.6.2.12 TRVSpellChecker.RegisterAsService, UnregisterAsService

Allows using this spelling checker as spelling checking service for a FireMonkey platform (IFMXSpellCheckerService, as well as IFMXSpellCheckerExService for Delphi 13+).

```
function RegisterAsService: Boolean;
procedure UnregisterAsService;
```

At startup, our component registers a platform spelling checking service that does it work by calling the default FireMonkey service (if available; otherwise, it marks all words as valid).

After calling **RegisterAsService**, this component will be used by our platform service to provide suggestions for misspelled words and word completions.

In other words, until you call **RegisterAsService**, our platform service works as a proxy for the previous platform service; after the call, it works as a proxy for this spelling checker.

UnregisterAsService is optional: the component automatically un-registers itself from the service on destruction.

Advantages of registering our spelling checker as a FireMonkey platform service:

- spelling checking on platforms where FireMonkey service is not implemented by Embarcadero yet
- using additional features (such as language selection)

6.2.6.2.13 TRVSpellChecker.RegisterEditor, UnregisterEditor

Registers/unregisters the specified editor in this spelling checker.

```
procedure RegisterEditor(Edit: TCustomRVControl);
procedure UnregisterEditor(Edit: TCustomRVControl);
```

These methods are implemented for the convenience, they are optional.

If you register an editor, the spell-checker:

- assigns the editor's OnSpellingCheck event to process spelling check in a background thread;
- calls the editor's ClearLiveSpellingResults when changing the active spelling checking language;
- calls the editor's LiveSpellingValidateWord when the user chooses “Add” or “Ignore All” in the dialog.

Additionally, you can call StartLiveSpelling to start a background spelling checking in all registered editors.

Input parameter:

Edit – TRichView (TRichView, TRichViewEdit, TDBRichView, TDBRichViewEdit) or ScaleRichView (TSRichViewEdit, TDBSRichViewEdit) control.

UnregisterEditor is usually not necessary: the control is automatically unregistered on destruction.

Compatibility: these methods require Delphi 6 and newer.

6.2.6.2.14 TRVSpellChecker.StartLiveSpelling

Starts a background spelling check in all registered editors.

```
procedure StartLiveSpelling;
```

This method calls StartLiveSpelling for a registered editors.

6.2.6.2.15 TRVSpellChecker.Suggest

Retrieves spelling suggestions for the specified word.

```
function Suggest(const AWord: TRVUnicodeString[1099]):
  TRVStringList[1041];
```

This method returns *nil* if there are no suggestions, or if a spell checker service is not available.

Note: some spellcheckers (for example, for Windows) can offer suggestions even for valid words.

6.2.6.2.16 TRVSpellChecker.SupportsFeatures

Defines an appearance of a spelling checking dialog.

```
function SupportsFeatures: TRVSpellFormActions[1093];
```

Value	Windows	macOS	iOS	Linux	Android
rvsfaIgnore	+	+	+	+	
rvsfaIgnoreAll	+	+	+	+	
rvsfaChange	+	+	+	+	
rvsfaChangeAll (auto-replacement and auto-correction)	+				
rvsfaAdd	+	+	+		
rvsfaGuessCompletions		+	+		

Note: do not use this function to check availability of spelling checker; use IsAvailable[735].

6.2.6.3 Events

In TRVSpellChecker

- OnSpellForm[737]
- OnSpellFormAction[738]
- OnSpellFormLocalize[739]

6.2.6.3.1 TRVSpellChecker.OnSpellForm

Occurs when a spelling checking dialog is created.

```
type
  TRVSpellFormEvent = procedure (
    Sender: TRVSpellChecker[728];
    Form: TfrmRVCustomSpell_) of object;

property OnSpellForm: TRVSpellFormEvent
```

You can perform additional form initializations in this event.

VCL and LCL

TfrmRVCustomSpell_ is a parent class of forms used as spelling checking dialogs (RVSpellFormUtils unit).

The actual dialogs are inherited from this class:

- standard forms:
  - TfrmRVSpell (RVSpellFrm unit), used if SpellFormStyle[732] = *rvsfsClassic*)
  - TfrmRVSpellWord (RVSpellWordFrm unit), used if SpellFormStyle = *rvspellfrmMSWord*)
- if TNT Controls (TMS Unicode components) are used:
  - TfrmRVTntSpell (RVTntSpellFrm unit), used if SpellFormStyle = *rvsfsClassic*)
  - TfrmRVTntSpellWord (RVTntSpellWordFrm unit), used if SpellFormStyle = *rvspellfrmMSWord*)

6.2.6.3.2 TRVSpellChecker.OnSpellFormAction

Occurs when the user has made a choice in the spelling checking dialog.

```
type
  TRVSpellFormActionEvent = procedure(
    Sender: TRVSpellChecker[728];
    const AWord, AReplaceTo: TRVUnicodeString[1099];
    AAction: TRVSpellFormAction[1093]) of object;

property OnSpellFormAction: TRVSpellFormActionEvent;
```

This event occurs when Execute[733] is called, a dialog window is shown, and the user made a choice in this dialog.

AAction	Meaning
rvsfaIgnore	The used decided to ignore a single occurrence of **AWord**
rvsfaIgnoreAll	The used decided to ignore all occurrences of **AWord**
rvsfaChange	The used decided to change one occurrence of **AWord** to **AReplaceTo**.
rvsfaChangeAll	The used decided to change all occurrences of **AWord** to **AReplaceTo**.
rvsfaAdd	The used decided to add **AWord** to the dictionary.

These operations will be performed by the spell-checker itself. You may perform additional actions in this event.

If you registered all checked editors using RegisterEditor[736] method, you do not need to perform additional actions. Otherwise, if **AAction** in [*rvsfaIgnoreAll, rvsfaAdd*], you should call LiveSpellingValidateWord[326](**AWord**) for all editors.

6.2.6.3.3 TRVSpellChecker.OnSpellFormLocalize

Occurs when a spelling checking dialog is created.

```
type
  TRVSpellFormEvent = procedure (
    Sender: TRVSpellChecker[728];
    Form: TfrmRVCustomSpell_) of object;

property OnSpellFormLocalize: TRVSpellFormEvent
```

This event is used by RichViewActions (TRVSpellInterface component) to localize controls in the dialog.

6.2.7 TRVStyle

This invisible at run-time component is used for defining a visual appearance of RichView[216] and its descendant components.

Unit [VCL/FMX] RVStyle / fmxRVStyle.

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[260] 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[216], RichViewEdit[469], DBRichView[604], DBRichViewEdit[616] controls.

But if you use the following:

- TCustomRichView.DeleteUnusedStyles[293];
- TDBRichViewEdit.AutoDeleteUnusedStyles[621]=*True*;
- TCustomRichView.LoadRVF[336], LoadRVFFromStream[335], and RVFTextStylesReadMode[258] or RVFParaStylesReadMode[258] <> *rvf_sIgnore*;
- TDBRichView or TDBRichViewEdit, and RVFTextStylesReadMode[258] or RVFParaStylesReadMode[258] <> *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[764], ParaStyles[757], or ListStyles[755];
- adds “Convert Lengths to twips” (... to pixels, ... to EMU) item in the context menu, allowing to convert all RVStyle properties measured in Units[764].
- adds 3 items into the context menu for component: “Edit Style Templates”, “Save Style Templates...”, “Load Style Templates...” which invoke property editors for StyleTemplates[762], 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[764] – collection of text attributes (styles), a collection of TFontInfo[821] items;
- ParaStyles[757] – collection of paragraph attributes (styles), a collection of TParaInfo[837] items;
- ListStyles[755] – collection of paragraph list styles (bullets and numbering), a collection of TRVListInfo[841] items.

Style templates ("real" styles)

- StyleTemplates[762] – collection of "real" styles, controlling properties of TextStyles[764] and ParaStyles[757];

Units of measurement

- Units[764] defines measure units for properties of this component and documents in linked TRichViiew controls.
- UnitsPixelsPerInch[765] defines DPI for values measured in pixels.

RVStyle allows to set some colors for associated controls:

- Color[744] – background color;
- SelColor, InactiveSelColor[751] – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor[751] – colors of selected text (with and without focus);
- DisabledFontColor[746] – text color for disabled controls.
- HoverColor[750] – default color of hypertext links under the mouse pointer;
- CheckpointColor, CheckpointEvColor[743] – colors of *checkpoints* (if visible);
- PageBreakColor[756] – color of explicit page breaks (if visible);
- FloatingLineColor[747] – color of placeholders for left- and right-aligned items (if visible).
- GridColor, GridReadOnlyColor[750] – colors of table grid lines (if visible); GridStyle, GridReadOnlyStyle[750] define their pen styles.

RVStyle allows to set cursors:

JumpCursor[752] – hypertext cursor for *hotspots*[178] and *hot-pictures*[172] (and default cursor for text hypertext items);
LineSelectCursor[753] – cursor for the left margin.

Tabs:

- SpacesInTab[761] – 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[168] is used;
- DefTabWidth[745] – default distance between tab stops.

Properties related to label items[181] and item types inherited from them:

- FieldHighlightColor[746] – color for highlighting label items.
- FieldHighlightType[747] specifies when to highlight label items.
- FootnoteNumbering[746] – numbering type for footnotes.
- FootnotePageReset[749] – numbering mode for footnotes.
- EndnoteNumbering[746] – numbering type for endnotes.
- SidenoteNumbering[746] – numbering type for sidenotes.

Properties related to selection:

- SelectionStyle[759] – visual appearance of selection (draw for objects or lines);
- SelectionMode[758] – modes of making selection;
- SelectionHandleKind[757] – visual appearance of touch screen selection handles;
- SelColor, InactiveSelColor[751] – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor[751] – colors of selected text (with and without focus).
- SelOpacity[751] – selection fill opacity (opaque and semitransparent selection is drawn differently)

Other properties

- TextEngine[763] specifies which API is used for drawing and measuring text;
- ParaBackAroundContent[757] specifies how boundaries of a paragraphs' background areas are determined;
- FontQuality[748] – font quality and anti-aliasing;
- DefCodePage[745] – default code page for internal ANSI <->Unicode conversion;
- LineWrapMode[753] specifies a line wrapping algorithm;
- UseSound[766].

Using main TRVStyle

If another TRVStyle component is assigned to MainRVStyle[756] property, most properties of this TRVStyle simply provide access to properties of this main TRVStyle.

(For example, SelColor[751] returns MainRVStyle[756].SelColor[751]).

Initially, this feature was implemented to allow using a single StyleTemplates[762] collection in multiple TRVStyle components. But in the new version of TRVStyle all properties of MainRVStyle are shared.

The following properties are not shared:

- TextStyles[764]
- ParaStyles[757]
- ListStyles[755]
- all events

Methods

You can save and load all information in RVStyle in ini-file:

- SaveINI[773] saves RVStyle properties in ini-file;
- LoadINI[771] loads RVStyle properties from ini-file;

or registry:

- SaveReg[773] saves RVStyle properties in ini-file;
- LoadReg[771] loads RVStyle properties from ini-file.

You can export styles in style sheet (for saving HTML files):

- SaveCSS[772];
- SaveCSSToStream[772].

You can search for a style with the specified properties:

- FindTextStyle, FindParaStyle[771]

Events

Custom drawing:

- OnDrawCheckpoint[776] – drawing checkpoint;
- OnDrawPageBreak[778] – drawing page break;
- OnDrawParaBack[780] – drawing background of paragraph;
- OnDrawTextBack[783] – drawing background of text;
- OnApplyStyle[774] – applying text style to Canvas;
- OnApplyStyleColor[775] – applying text style color to Canvas;
- OnDrawStyleText[781] – drawing text of the given style;
- OnStyleHoverSensitive[784] – "should the component be repainted when user moves the mouse pointer over hyperlink of the given style?".

6.2.7.1 Properties

In TRVStyle

- CheckpointColor[743]
- CheckpointEvColor[743]
- Color[744]
- CurrentItemColor[744]
- DefCodePage[745]
- DefTabWidth[745]
- DisabledFontColor[746]
- EndnoteNumbering[746]
- FieldHighlightColor[746]
- FieldHighlightType[747]
- FloatingLineColor[747]
- FontQuality[748] [VCL and LCL]
- FootnoteNumbering[746]
- FootnotePageReset[749]
- FullRedraw[750] [VCL and LCL]
- GridColor[750]
- GridReadOnlyColor[750]
- GridStyle[750]
- GridGridReadOnlyStyle[750]
- HoverColor[750]
- InactiveSelColor[751]
- InactiveSelTextColor[751]
- InvalidPicture[752]
- JumpCursor[752]
- LineSelectCursor[753]

- LineWrapMode [753]
- ListStyles [755]
- LiveSpellingColor [755]
- MainRVStyle [756]
- PageBreakColor [756]
- ParaBackAroundContent [757]
- ParaStyles [757]
- SelColor [751]
- SelectionHandleKind [757] (D2010+)
- SelectionMode [758]
- SelectionStyle [759]
- SelOpacity [751]
- SelTextColor [751]
- SidenoteNumbering [746]
- SoftPageBreakColor [761]
- SpacesInTab [761]
- StyleTemplates [762]
- TextBackgroundKind [762]
- TextEngine [763] [VCL and LCL]
- TextStyles [764]
- Units [764]
- UnitsPixelsPerInch [765]
- UseSound [766]

6.2.7.1.1 TRVStyle.CheckpointColor, CheckpointColor

A color of dotted lines showing positions of *checkpoints* in document.

```
property CheckpointColor: TRVColor [1063];
property CheckpointEvColor: TRVColor [1063];
```

If TRichView shows its *checkpoints* (*rvoShowCheckpoints* in TRichView.Options [247]), 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 [776] event.

TRichView must be linked with this TRVStyle (see TRichView.Style [260]).

If MainRVStyle [756] is assigned, these properties provide access to the corresponding properties of MainRVStyle [756].

Default value

CheckpointColor: rvclGreen [1105]
CheckpointEvColor: rvclLime [1105]

See also events:

- OnDrawCheckpoint [776].

See also properties of TCustomRichView:

- Options[247].

See also:

- *Checkpoints* overview[92].

6.2.7.1.2 TRVStyle.Color

A default background color for TRichView component linked to this TRVStyle component.

```
property Color: TRVColor[1063];
```

This property may be overridden with TRichView.Color[233] property.

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value for VCL and LCL:

clWindow

Default value for FireMonkey:

TAlphaColorRec.Null (a transparent document background, so a styled component background is visible, unless DarkMode[234]).

See also properties of TCustomRichView:

- Style[260];
- Color[233];
- BackgroundProperties[229];
- BackgroundPicture[229].

6.2.7.1.3 TRVStyle.CurrentItemColor

A color for drawing frame around the current item in TRichViewEdit[469].

```
property CurrentItemColor: TRVColor[1063];
```

(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[260]).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclNone[1105]

6.2.7.1.4 TRVStyle.DefCodePage

A default code page used inside TRichView for automatic conversions of ANSI strings to Unicode and vice versa

```
property DefCodePage: TRVCodePage[1062];
```

(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[809] 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.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

0 (Windows default code page, *CP_ACP*)

See also:

- Unicode in RichView[135].

6.2.7.1.5 TRVStyle.DefTabWidth

A default distance between tab stops

```
property DefTabWidth: TRVStyleLength[1094];
```

(introduced in v1.9)

This value is measured Style[260].Units[764].

Tab stops are started from the LeftMargin[245] (or from the RightMargin[251] for right-to-left[137] paragraphs).

Note: tabulators[168] can be inserted from text/RTF files or keyboard only if value of SpacesInTab[761] property is 0 (otherwise tabs are inserted as several space characters).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

48

See also properties:

- ParaStyles[757].Tabs[836].

See also properties:

- Units of measurement in TRichView[144]

6.2.7.1.6 TRVStyle.DefUnicodeStyle

```
property DefUnicodeStyle: Integer;
```

Obsolete property. Not available in FireMonkey version.

Default value:

-1

6.2.7.1.7 TRVStyle.DisabledFontColor

Defines a font color to use in disabled TRichView[216] controls.

```
property DisabledFontColor: TRVColor[1063];
```

(Introduced in version 13)

If this property = *rvclNone*[1105], a disabled control uses the same text color as enabled control.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclNone[1105]

6.2.7.1.8 TRVStyle.EndnoteNumbering

A numbering type for endnotes[184], footnotes[185], and sidenotes[186].

```
property EndnoteNumbering: TRVSeqType
property FootnoteNumbering: TRVSeqType
property SidenoteNumbering: TRVSeqType
```

(introduced in version 10 and 15)

See TRVSeqType[1092] for possible values.

If you changed value of these properties while the document in the linked[260] TRichView[216] control is displayed, call RichView.RefreshSequences[340] and RichView.Format[295].

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default values:

- EndnoteNumbering: *rvseqLowerRoman*
- FootnoteNumbering: *rvseqDecimal*
- SidenoteNumbering: *rvseqLowerAlpha*

See also:

- FootnotePageReset[749].

6.2.7.1.9 TRVStyle.FieldHighlightColor

A color for highlighting labels[181], numbered sequences[183], endnotes[184], footnotes[185], references to parent footnotes and endnotes[190].

```
property FieldHighlightColor: TRVColor[1063];
```

(introduced in version 10)

See FieldHighlightType[747] for details.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclBtnFace[1105]

6.2.7.1.10 TRVStyle.FieldHighlightType

Specifies when FieldHighlightColor[746] must be used for highlighting labels[181], numbered sequences[183], endnotes[184], footnotes[185], references to parent footnotes and endnotes[190].

```
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[486] property must be *True*.
rvfhAlways	All items are always highlighted.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvfhCurrent

6.2.7.1.11 TRVStyle.FloatingLineColor

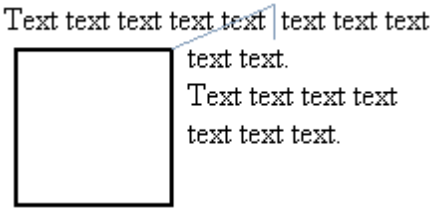
Color for displaying places of insertion for left- and right-aligned[1100] items, and markers for text box items[188].

```
property FloatingLineColor: TRVColor[1063];
```

(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.



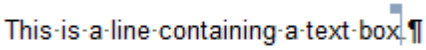
The lines are displayed in the following cases:

- in TCustomRichViewEdit[469], for one item, when this item is a current item (i.e. it is at the position of the caret)
- for all side-aligned items, if *rvoShowSpecialCharacters* is included in Options[247], and *rvcFloatingLines* is included in RVVisibleSpecialCharacters[1129]

Assign *rvclNone*[1105] to prevent displaying these lines.

Text box items

A special mark is inserted in places of insertion of text box items[188]:



The marks are displayed in the following cases:

- in TCustomRichViewEdit[469], 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[247], and *rvcPlaceholders* is included in RVVisibleSpecialCharacters[1129].

If FloatingLineColor=*rvclNone*[1105], SpecialCharactersColor[762] is used. If SpecialCharactersColor=*rvclNone*[1105], *rvclBtnShadow*[1105] is used.

ScaleRichView note: in TSRichViewEdit, this color is also used to draw lines to places of insertion of sidenotes[186] and text box items, and around the edited floating box. Assign *clNone* to prevent displaying these lines.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value (VCL and LCL / FireMonkey):

$BDA28A / $FF8AA2BD

6.2.7.1.12 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.

Type	Meaning
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.

Note: This property is not available in FireMonkey

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

fqDefault

6.2.7.1.13 TRVStyle.FootnotePageReset

Specifies whether to reset numbering of footnotes[185] 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[253].IgnoreNotes[463]=*False*.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

True

See also:

- FootnoteNumbering[746]

6.2.7.1.14 TRVStyle.FullRedraw

```
property FullRedraw: Boolean;
```

Obsolete property. Not available in FireMonkey version.

Default value:

False

6.2.7.1.15 TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle

Colors and styles of table [179] grid lines.

```
property GridColor: TRVColor[1063];
property GridReadOnlyColor: TRVColor[1063];
property GridStyle: TRVPenStyle[1084];
property GridReadOnlyStyle: TRVPenStyle[1084];
```

(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 [216] or in TRichViewEdit [469] in read-only [488] mode.

Table borders can be invisible in the following cases:

- their width is 0 (see BorderWidth [909] and CellBorderWidth [910] properties);
- some sides are hidden (see VisibleBorders [921] and Cell [951].VisibleBorders [963] 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 [247].

You can hide/show table grid for the specific table, see *rvtoHideGridLines* in Table.Options [916] property.

TRichView must be linked with this TRVStyle (see TRichView.Style [260]).

If MainRVStyle [756] is assigned, these properties provide access to the corresponding properties of MainRVStyle [756].

Default values:

- GridColor: *rvclBtnFace* [1105];
- GridReadOnlyColor: *rvclBtnFace* [1105];
- GridStyle: *psDot (VCL and LCL) / rvpsSolid* [1110] *(FireMonkey)*
- GridReadOnlyStyle: *rvpsClear* [1110].

6.2.7.1.16 TRVStyle.HoverColor

Default a text hover color.

```
property HoverColor: TRVColor[1063];
```

A text hover color is a text color under the mouse pointer.

All hypertext styles have their own hover color (see TFontInfo.HoverColor [812]). If they are equal to *rvclNone* [1105], this **HoverColor** is used.

Set to *rvclNone* [1105] to disable hover color effect.

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:
rvclNone[1105]

6.2.7.1.17 TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor, SelOpacity

Color of text and background of selected text in TRichView components.

```
property SelColor: TRVColor[1063];
property SelTextColor: TRVColor[1063];
property InactiveSelColor: TRVColor[1063];
property InactiveSelTextColor: TRVColor[1063];
property SelOpacity: TRVOpacity[1082];
```

("Inactive" properties are introduced in version 1.6, SelOpacity is introduced in version 20)

If MainRVStyle[756] is assigned, these properties provide access to the corresponding properties of MainRVStyle[756].

Mode 1: Opaque selection drawing

This mode is used when **SelOpacity** = 100000 (meaning 100% opacity).

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*[176] and *hotspots*[178],
- drawing a frame around the selected *breaks*[173],
- drawing background for the selected pictures[169] and controls[174],
- drawing background for selected table[179] cells.

In FireMonkey, **SelColor** and **SelTextColor** are used for drawing resizing handles and a resizing rectangle (for pictures[169] and controls[174]), **SelColor** is used for drawing a table[179] sizing line.

Set **InactiveSelColor** and **InactiveSelTextColor** to *rvclNone*[1105] to hide a selection when the input focus shifts to another control.

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

Mode 2: Semitransparent selection drawing

This mode is used when **SelOpacity** < 100000 (meaning semitransparency).

In this mode, only **SelColor** and **InactiveSelColor** are used (**SelTextColor** and **InactiveSelTextColor** are ignored). Selection is shaded with **SelColor** (if the control is focused) or **InactiveSelColor** (if not focused).

Default selection drawing [FMX]

In FireMonkey, if *rvoDefaultSelectionFill* is included in the Options [247] property of TRichView, the component searches for 'selection' brush resource in its visual style (specified in StyleLookup property). If found, a color and an opacity of this brush resource is assigned to **SelColor** and **SelOpacity**; **InactiveSelColor** is assigned to the same color as **SelColor** but more transparent.

Note: by default, TRichView is linked with 'memostyle' visual style, and this style contains semitransparent brush resource named 'selection'.

Default values:

- SelColor: *rvclHighlight* [1105]
- SelTextColor: *rvclHighlightText* [1105]
- InactiveSelColor: *rvclHighlight* [1105]
- InactiveTextColor: *rvclHighlightText* [1105]
- SelOpacity: 100000 (opaque)

See also properties:

- SelectionStyle [759].

6.2.7.1.18 TRVStyle.InvalidPicture

Picture that will be inserted when reading RTF or RVF containing invalid or missing pictures.

```
property InvalidPicture: TRVPicture[1037];
```

(Introduced in version 1.7)

Default value:

(TBitmap)

If MainRVStyle [756] is assigned, this property provides access to the corresponding property of MainRVStyle [756].

See also properties of TCustomRichView:

- RVFWarnings [258].

6.2.7.1.19 TRVStyle.JumpCursor

Mouse cursor for hypertext indication.

```
property JumpCursor : TCursor
```

This cursor appears when user moves the mouse pointer to *hotspot* [178] or *hot-picture* [172] 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 [813]).

TRichView must be linked with this TRVStyle (see TRichView.Style [260]).

If MainRVStyle [756] is assigned, this property provides access to the corresponding property of MainRVStyle [756].

Default value:

crJump

See also:

- Cursors used by the components of TRichView family[147].

6.2.7.1.20 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[230].

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

You can disable line selection if you set this property to *crNone* (-1).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

crRVFlipArrow

See also properties of TCustomRichView:

- LeftMargin[245].

See also:

- Cursors used by the components of TRichView family[147].

6.2.7.1.21 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[167] are added using line breaking rules. Line breaks between text items[167] 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[135] text items.

For ANSI text, line breaks occur only on space characters.

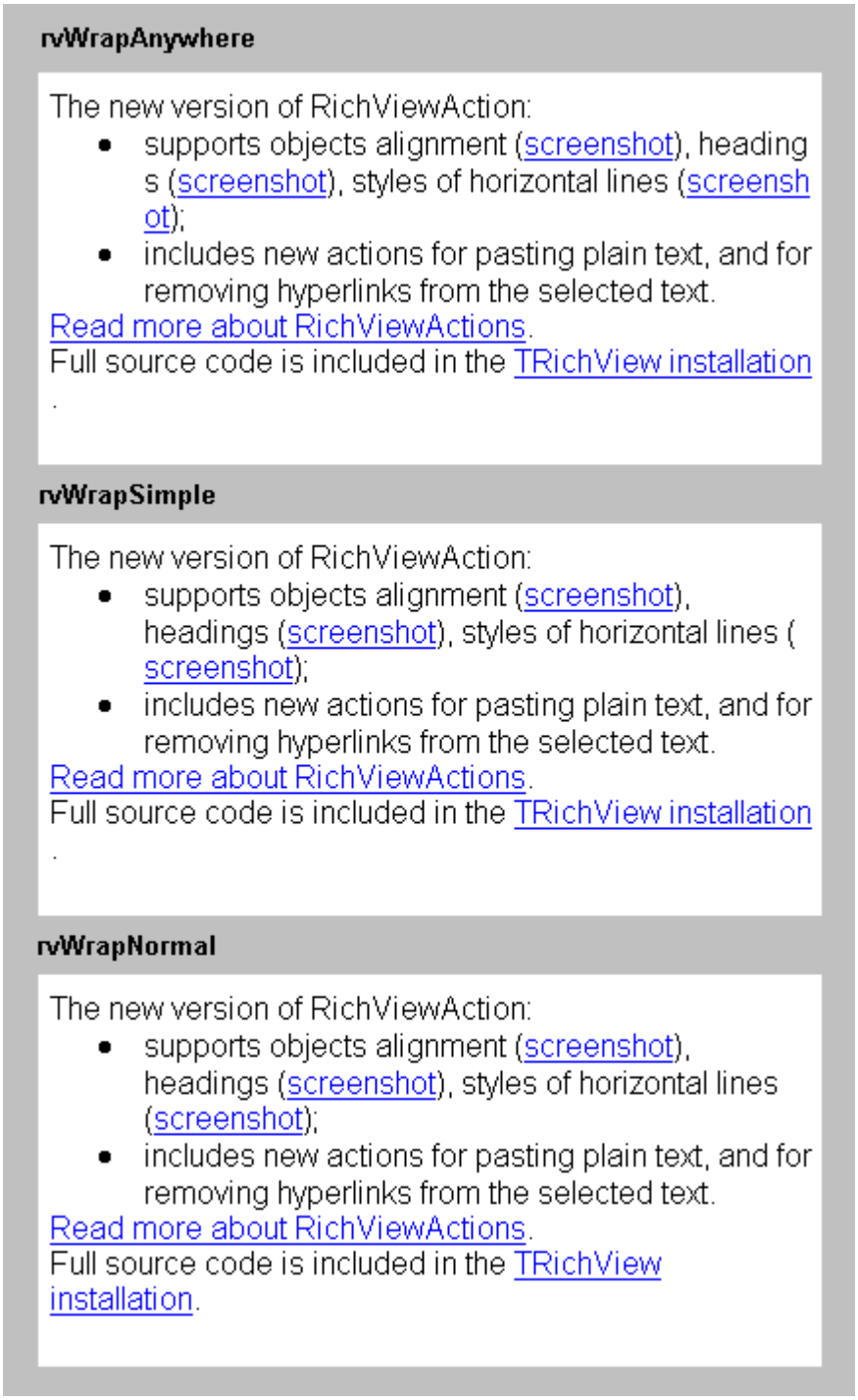
For Unicode text, line breaks are added using advanced rules. For example, in the string '(a)(b)' the only possible line break is between ')' and '('.

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



If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvWrapNormal

6.2.7.1.22 TRVStyle.ListStyles

This is one of the key properties of TRVStyle: collection of paragraph list styles (bullets and numbering).

```
property ListStyles : TRVListInfos[795];
```

Bullets/numbering (lists) are implemented in the following way:

1. TRVStyle has a collection of list styles (this property). Each list style[841] contains a collection of list levels (Levels[842] property). A list level[859] defines the most of list properties.
2. List markers[180] 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[764];
- ParaStyles[757].

See also types:

- TRVListInfos[795] (collection of list styles);
- TRVListInfo[841] (item in the collection of list styles).

See also methods of TCustomRichView:

- SetListMarkerInfo[369],
- GetListMarkerInfo[315].

See also methods of TCustomRichViewEdit:

- ApplyListStyle[498],
- RemoveLists[552],
- ChangeListLevels[508].

See also:

- List markers item type[180].

6.2.7.1.23 TRVStyle.LiveSpellingColor

Color of live spelling's wavy underlines.

```
property LiveSpellingColor: TRVColor[1063];
```

(introduced in v1.9)

This color is used for underlines that mark misspelled words.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Example:

Misspeled text

Default value:

rvclRed[1105]

See also:

- Live spelling check in TRichView[138].

6.2.7.1.24 TRVStyle.MainRVStyle

Defines the master TRVStyle component for this component.

```
property MainRVStyle: TRVStyle[739];
```

(introduced in version 14; meaning is extended in version 20)

If this **MainRVStyle** is assigned, almost all properties of this TRVStyle object provide access to corresponding properties of **MainRVStyle**.

The following properties are not affected:

- TextStyles[764]
- ParaStyles[757]
- ListStyles[755]
- all events

This property allows:

- several editors can work with the same collection of style templates[762]
- to simplify assignment to other properties (a group of linked TRVStyle objects shares properties, so it is enough to assign a property of any of them)

If some TRichView component is used as a header[363]/footer[362] (and style templates are used[263]) it must have Style[260].MainRVStyle assigned equal to Style[260] of the main editor.

ScaleRichView note: TSRichViewEdit provides that srve.RichViewEdit.Style = srve.RVHeader.Style.MainRVStyle = srve.RVFooter.Style.MainRVStyle.

6.2.7.1.25 TRVStyle.PageBreakColor

Color of explicit (user-defined) page breaks in TRichView components.

```
property PageBreakColor: TRVColor[1063];
```

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[778] event.

Explicit page breaks are added by assigning *True* to TRichView.PageBreaksBeforeItems[250][], or using TRichViewEdit.InsertPageBreak[534].

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclBtnShadow[1105]

See also properties:

- SoftPageBreakColor[761].

See also events:

- OnDrawPageBreak[778].

See also properties of TCustomRichView:

- Options[247] (*rvoShowPageBreaks*).

6.2.7.1.26 TRVStyle.ParaBackAroundContent

Specifies how boundaries of a paragraphs' background areas are determined.

```
property ParaBackAroundContent: Boolean;
```

(introduced in version 24)

If *False*, the background area occupies the full available width, taking margins and indents into account.

If *True*, the background is drawn only around the content. This mode is used by TRVChat[640].

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value

False

6.2.7.1.27 TRVStyle.ParaStyles

This is one of the key properties of TRVStyle: collection of paragraph styles.

```
property ParaStyles : TParaInfos[791];
```

Paragraph styles define paragraph attributes in documents.

See also properties:

- TextStyles[764];
- ListStyles[755].

See also types:

- TParaInfos[791] (collection of paragraph styles);
- TParaInfo[828] (item in collection of paragraph styles).

See also:

- Paragraphs in RichView[100].

6.2.7.1.28 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.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

FireMonkey note: these selection handles are used only if FireMonkey style used in TRichView control does not contain selection handles (TSelectionPoint objects named 'leftselectionpoint' and 'rightselectionpoint'). If a style contains selection handles, they are used instead.

Default value:

rvshkWedge

See also properties of TCustomRichView:

- SelectionHandlesVisible[260]

6.2.7.1.29 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 Mode	Meaning
	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.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvsmWord

6.2.7.1.30 TRVStyle.SelectionStyle

Defines visual appearance of selection in TRichView components.

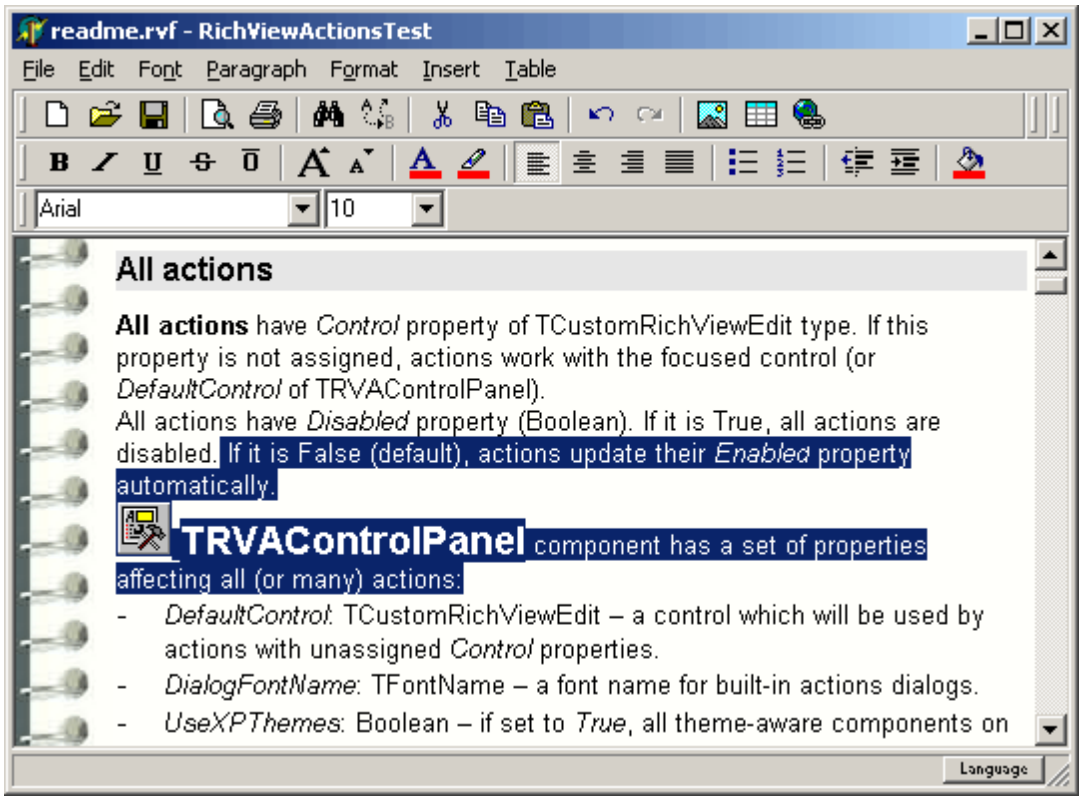
```
property SelectionStyle: TRVSelectionStyle;

type
  TRVSelectionStyle = (rvssItems, rvssLines);
```

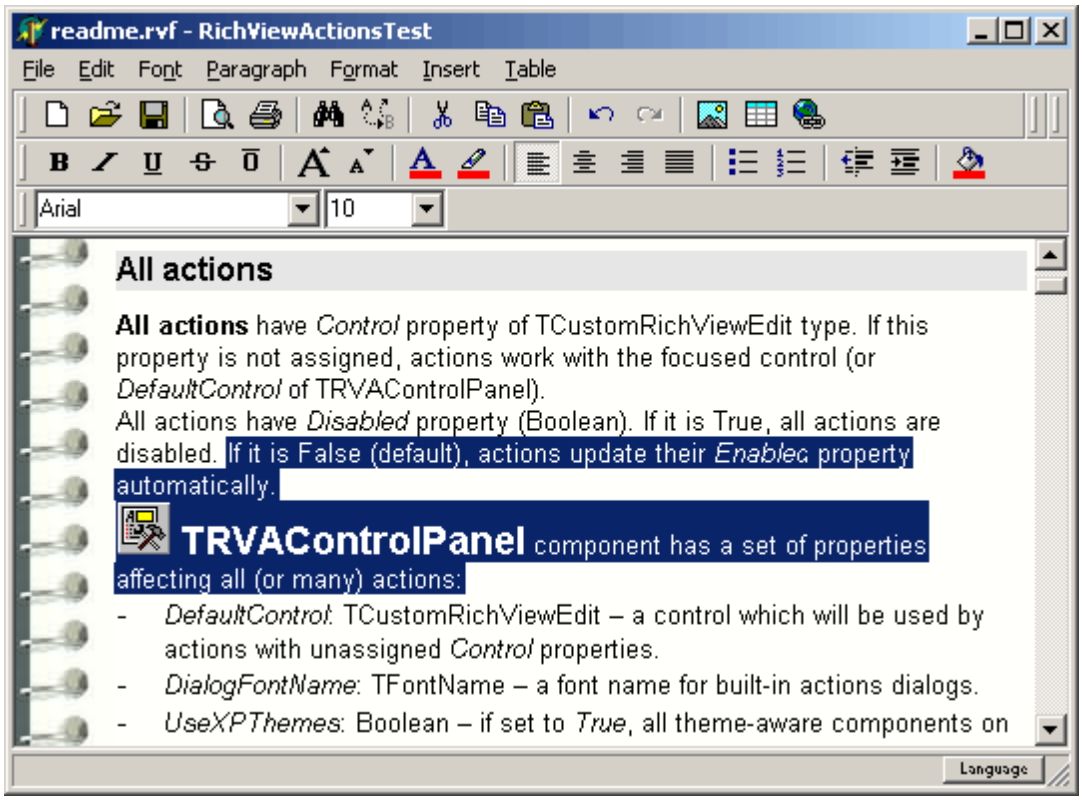
(introduced in v 1.8)

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[137] =*rvbdUnspecified*.

rvssItems:



rvssLines:



If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvssItems

See also properties:

- SelColor, SelTextColor, InactiveSelColor, InactiveSelTextColor[751];
- SelectionMode[758].

6.2.7.1.31 TRVStyle.SoftPageBreakColor

Color of soft page breaks in TRichView components.

```
property PageBreakColor: TRVColor[1063];
```

This color is used for drawing horizontal lines (with "folded paper" icon) specifying soft (automatic) page breaks. You can change this drawing in OnDrawPageBreak[778] event.

TRichView must be linked with this TRVStyle (see TRichView.Style[260]). Soft page breaks displaying is turned on by TRichView.AssignPageBreaks[284].

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclBtnFace[1105]

See also properties:

- PageBreakColor[756].

See also events:

- OnDrawPageBreak[778].

See also properties of TCustomRichView:

- Options[247] (*rvoShowPageBreaks*).

6.2.7.1.32 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[216] and TRichViewEdit[469] for adding/inserting multiline text and RTF, or on typing.

TRichView must be linked with this TRVStyle (see TRichView.Style[260]).

If this property is 0, TABs are inserted as a special tabulator item type[168].

Note: default value is changed to 0 since v1.9

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

0

See also properties:

- DefTabWidth[745].

6.2.7.1.33 TRVStyle.SpecialCharactersColor

Color of special characters.

```
property SpecialCharactersColor: TRVColor[1063];
```

(introduced in version 12)

if *rvoShowSpecialCharacters* is included in TRichView.Options[247], this color is used for drawing marks denoting space characters and paragraph breaks.

If it equals to *rvclNone*[1105], a normal text color is used instead.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvclNone[1105]

See also:

- RVVisibleSpecialCharacters[1129] global typed constant
- FloatingLineColor[747]

6.2.7.1.34 TRVStyle.StyleTemplates

A collection of real styles, controlling text attributes from TextStyles[764] and paragraph attributes from ParaStyles[757].

```
property StyleTemplates : TRVStyleTemplateCollection[797];
```

Using style templates is optional, they are used only for richview controls having UseStyleTemplates[263]=*True*.

If MainRVStyle[756] is assigned, this property returns MainRVStyle[756].**StyleTemplates**.

See also

- Styles and style templates[103].

6.2.7.1.35 TRVStyle.TextBackgroundKind

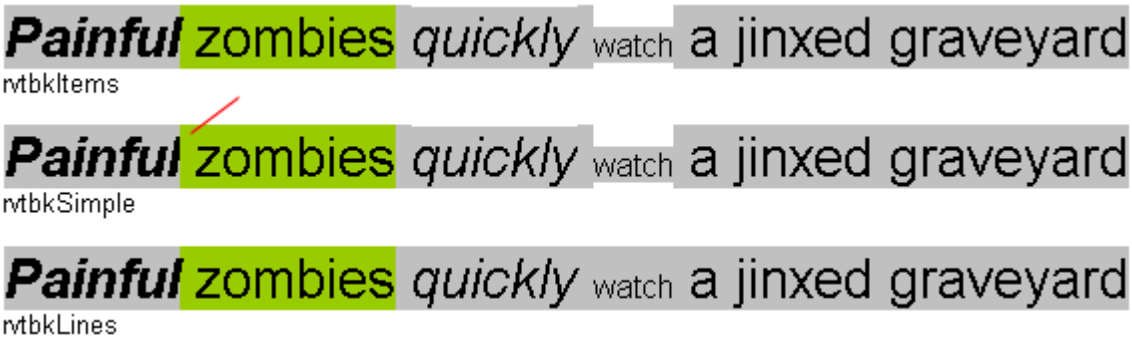
Defines visual appearance of text background in TRichView components.

```
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).

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Default value:

rvtbkItems

See also:

- OnDrawTextBack[783]

See also properties of text style:

- BackColor[808]
- HoverBackColor[811];

6.2.7.1.36 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 bidirectional text is turned off. **Cons:** • completely incorrect results for right-to-left text (Arabic and Hebrew), if the support of bidirectional text is turned off; • results may be not completely correct if text includes some "exotic" characters; • unreliable results when the support of bidirectional text is turned on (may be correct for one font, but incorrect for another font).

Value	Meaning	Comments
rvteUniscribe	***Uniscribe***	**Pros:** • reliable processing of complex scripts and "exotic" characters; • acceptable results for right-to-left text even if support of bidirectional text is turned off **Cons:** • usually slower

Reformat documents in linked TRichView controls after changing value of this property.

If MainRVStyle[756] is assigned, this property provides access to the corresponding property of MainRVStyle[756].

Note: this property is not available in FireMonkey version.

Default value:

rvteUniscribe

6.2.7.1.37 TRVStyle.TextStyles

This is one of key properties of TRVStyle: a collection of text attributes (styles).

```
property TextStyles : TFontInfos[785];
```

Text styles define fonts and other text attributes in documents.

The first text style in the collection (TextStyles[0]) has some additional meaning:

- *breaks*[90] with color = *clNone* (default value) have the color of this style;
- Charset[809] 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[815] for this style;
- this style can be hypertext, but it is not recommended.

See also properties:

- ParaStyles[757];
- ListStyles[755].

See also types:

- TFontInfos[785] (collection of text styles);
- TFontInfo[821] (item in collection of text styles).

6.2.7.1.38 TRVStyle.Units

Defines units of measurement for properties of TRVStyle and linked[260] TRichView controls.

```
property Units: TRVStyleUnits[1094]
```

(introduced in version 13)

All values of type TRVStyleLength[1094] are measured in units specified in this property.

If Units = *rvstuPixels*, DPI (dots per inch) of values is defined n UnitsPixelsPerInch[765] property (i.e. 1 pixel = 1/UnitsPixelsPerInch of an inch).

Assignment to this property does not convert properties to new units. To change units of measurement, at first call one of ConvertDocTo* [287] methods of all TRichView [216] components linked [260] to this TRVStyle, then call one of ConvertTo* [767] 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 Lengths to Pixels", "Convert Lengths to Twips", "Convert Lengths to EMU" commands).

If MainRVStyle [756] is assigned, this property provides access to the corresponding property of MainRVStyle [756].

Default value:

rvstuPixels

See also properties:

- Units of measurement in TRichView [144] (contains a list of properties measured in Units)

6.2.7.1.39 TRVStyle.UnitsPixelsPerInch

Specifies number of pixels in one inch.

```
property UnitsPixelsPerInch: Integer;
```

(introduced in version 18)

This property is applied to:

- values (of TRVStyleLength [1094] type) measured in Units [764]: TRVStyleUnits [1094], if Units = *rvstuPixels*;
- values (of TRVLength [1081] type) measured in TRVUnits [1099] , if Units = *rvuPixels*;
- values measured in pixels (of TRVPixelLength [1085] type);
- sizes of pictures [169] in documents (if they are not resized by the user).

When displaying these sizes on a device (screen or paper), they are recalculated according to the device resolution, so these sizes are device-independent.

It's highly recommended to leave the default value of this property (96). With this setting, pixels in TRichView are similar to pixels in HTML (which also assume 96 DPI).

Alternatively, you can assign a value of Screen.PixelsPerInch to this property, to measure values in screen pixels (as it was default in TRichView versions prior to 18).

You can implement zooming using this property (for example, if UnitsPixelsPerInch = 48, value = 100 is displayed in 96 DPI as 100 * 96 / 48 = 200, i.e. 200% zoom). However, it is highly not recommended, because:

- this setting affects everything (file import and export, printing), not only displaying ion the screen;
- this setting does not affect text size.

The recommended way of zooming is changing TRichView [216].DocumentPixelsPerInch [239] property.

If MainRVStyle [756] is assigned, this property provides access to the corresponding property of MainRVStyle [756].

RichViewActions note

If you use TRVRuler from RichViewActions, after changing value of UnitsPixelsPerInch, you need to change the ruler's UnitsPixelsPerInch accordingly:

```
RVRuler1.UnitsPixelsPerInch :=
  RVStyle1.UnitsPixelsPerInch;
```

The same for TrvActionPageSetup:

```
rvActionPageSetup.UnitsPixelsPerInch :=
  RVStyle1.UnitsPixelsPerInch;
```

Default value:

96

See also:

- Units of measurement in TRichView [144]

6.2.7.1.40 TRVStyle.UseSound

Defines whether RichViewEdit [469] will use sound.

```
property UseSound: Boolean
```

(introduced in version 1.3)

If *True*, RichViewEdit plays a system default sound when user tries to modify a read-only [488] or a protected [815] text.

If MainRVStyle [756] is assigned, this property provides access to the corresponding property of MainRVStyle [756].

Default value:

True

6.2.7.2 Methods

In TRVStyle

AddTextStyle [767] (deprecated)
Create [767]
ConvertToPixels, ConvertToTwips, ConvertToEMU, ConvertToDifferentUnits [767]
DeleteTextStyle [767] (deprecated)
Destroy [767]
GetAsPixels, GetAsStandardPixels, GetAsHTMLPixels, GetAsTwips, GetAsDifferentUnits, GetAsRVUnits [768]
PixelsToUnits, StandardPixelsToUnits, HTMLPixelsToUnits, TwipsToUnits, DifferentUnitsToUnits, RVUnitsToUnits [769]
FindParaStyle, FindTextStyle [771]
GetListStyleClass [770]
GetParaStyleClass [770]
GetTextStyleClass [770]
LoadINI [771]
LoadReg [771]
SaveCSS [772]
SaveCSSToStream [772]
SaveINI [773]
SaveReg [773]

6.2.7.2.1 TRVStyle.AddTextStyle

```
function AddTextStyle: Integer;
```

Obsolete. In order to add item in collection of text styles, use RVStyle.TextStyles.Add[787]

6.2.7.2.2 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.

6.2.7.2.3 TRVStyle.ConvertTo*

The methods convert all properties of this TRVStyle component to new units of measurement.

```
procedure ConvertToPixels;
procedure ConvertToTwips;
procedure ConvertToEMU;
procedure ConvertToDifferentUnits(NewUnits: TRVStyleUnits[1094]);
```

(introduced in versions 13, 18)

The methods convert all properties of this TRVStyle (including all sub-properties) from Units[764] to:

- pixels (1 pixel = 1/UnitsPixelsPerInch[765] of an inch)
- twips,
- EMU
- NewUnits (pixels, twips, or EMU),

respectively.

After the conversion, a new value is assigned to Units[764] property.

If there is a document in TRichView control linked to this TRVStyle component, this document must be converted[287] before calling this method.

See also:

- Units of measurement in TRichView[144]

6.2.7.2.4 TRVStyle.DeleteTextStyle

```
procedure DeleteTextStyle(Index: Integer);
```

Obsolete. In order to delete the i-th item from text styles, use TRVStyle.TextStyles[764][i].Free

6.2.7.2.5 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.

6.2.7.2.6 TRVStyle.GetAs*

The methods return **Value** converted from Units [764] to different units of measurement.

```
function GetAsPixels(Value: TRVStyleLength[1094];
  PixelsPerInch, MinNonZeroValue: TRVCoord[1064]): TRVCoord[1064];
function GetAsHTMLPixels(Value: TRVStyleLength[1094]): TRVPixel96Length[1085];
function GetAsStandardPixels(Value: TRVStyleLength[1094]): TRVPixelLength[1085];

function GetAsTwips(Value: TRVStyleLength[1094]): Integer;
function GetAsEMU(Value: TRVStyleLength[1094]): Integer;
function GetAsDifferentUnits(Value: TRVStyleLength[1094];
  Units: TRVStyleUnits[1094]): TRVStyleLength[1094];
function GetAsRVUnits(Value: TRVStyleLength[1094];
  RVUnits: TRVUnits[1099]): TRVLength[1081];
```

(introduced in versions 13, 18)

Method	Converts to
GetAsPixels	Pixels. 1 pixel = 1/PixelsPerInch of an inch (where PixelsPerInch is specified as a parameter)
GetAsHTMLPixels	Pixels. 1 pixel = 1/96 of an inch.
GetAsStandardPixels	Pixels. 1 pixel = 1/UnitsPixelsPerInch [765] of an inch.
GetAsTwips	Twips. 1 twip = 1/20 of an inch.
GetAsEMU	EMU. 1 EMU = 1/914400 of an inch = 1/36000 mm.
GetAsDifferentUnits	Units (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch [765] is used as DPI.
GetAsRVUnits	RVUnits (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch [765] is used as DPI.

When GetAsPixels converts a nonzero Value to pixels, the result cannot be less than MinNonZeroValue (parameter) (or -MinNonZeroValue for a negative Value). For other methods converting nonzero Value to pixels, the result cannot be less than 1 (or -1 for a negative Value).

See also:

- *ToUnits [769] methods;

- ConvertTo* [767] methods;
- Units of measurement in TRichView [144]

6.2.7.2.7 TRVStyle.*ToUnits

The methods return **Value** converted from different units of measurement to Units [764].

```
function PixelsToUnits(Value: TRVCoord[1064];
  PixelsPerInch: Integer): TRVStyleLength[1094];
function HTMLPixelsToUnits(Value: TRVPixel96Length[1085]): TRVStyleLength[1094];
function StandardPixelsToUnits(
  Value: TRVPixelLength[1085]): TRVStyleLength[1094];

function TwipsToUnits(Value: Integer): TRVStyleLength[1094];
function EMUToUnits(Value: Integer): TRVStyleLength[1094];
function DifferentUnitsToUnits(Value: TRVStyleLength[1094];
  Units: TRVStyleUnits[1094]): TRVStyleLength[1094];
function RVUnitsToUnits(Value: TRVLength[1081];
  RVUnits: TRVUnits[1099]): TRVStyleLength[1094];
```

(introduced in versions 13, 18)

Method	Converts from
PixelsToUnits	Pixels. 1 pixel = 1/PixelsPerInch of an inch (where PixelsPerInch is specified as a parameter)
HTMLPixelsToUnits	Pixels. 1 pixel = 1/96 of an inch.
StandardPixelsToUnits	Pixels. 1 pixel = 1/UnitsPixelsPerInch [765] of an inch.
TwipsToUnits	Twips. 1 twip = 1/20 of an inch.
EMUToUnits	EMU. 1 EMU = 1/914400 of an inch = 1/36000 mm.
DifferentUnitsToUnits	Units (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch [765] is used as DPI.
RVUnitsToUnits	RVUnits (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch [765] is used as DPI.

When these methods convert a nonzero Value to pixels, the result cannot be less than 1 (or -1 for a negative Value).

See also:

- GetAs* [768] methods;
- ConvertTo* [767] methods;
- Units of measurement in TRichView [144].

6.2.7.2.8 TRVStyle.GetListStyleClass

Returns the class type representing a style of paragraph lists (bullets and numbering).

```
type
  TRVListInfoClass = class of TRVListInfo[841];

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 [841]) and use it in this descendant component.

See also:

- GetParaStyleClass [770];
- GetTextStyleClass [770].

6.2.7.2.9 TRVStyle.GetParaStyleClass

Returns the class type representing a style of paragraphs.

```
type
  TRVParaInfoClass = class of TParaInfo[828];

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 [828]) and use it in this descendant component.

See also:

- GetListStyleClass [770];
- GetTextStyleClass [770].

6.2.7.2.10 TRVStyle.GetTextStyleClass

Returns the class type representing a style of text.

```
type
  TRVFontInfoClass = class of TFontInfo[821];

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 [821]) and use it in this descendant component.

See also:

- GetListStyleClass [770];
- GetParaStyleClass [770].

6.2.7.2.11 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 [773];
- LoadReg [771].

6.2.7.2.12 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[821]): Integer;
function FindParaStyle(ParaStyle: TParaInfo[837]): Integer;
```

(introduced in version 14)

The methods are equivalent to:

```
TextStyles[764].FindSuchStyleEx[791](0, TextStyle, RVAllFontInfoProperties[1075])
ParaStyles[757].FindSuchStyleEx[794](0, ParaStyle, RVAllParaInfoProperties[1083])
```

6.2.7.2.13 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 [773];
- LoadINI [771].

6.2.7.2.14 TRVStyle.SaveCSS

Saves text and paragraph styles as CSS (Cascading Style Sheets) file.

```
type
  TRVSaveCSSOption = (
    rvcssOnlyDifference, rvcssIgnoreLeftAlignment,
    rvcssNoDefCSSStyle, rvcssDefault0Style,
    rvcssMarkersAsText);
  TRVSaveCSSOptions = set of TRVSaveCSSOption;

function SaveCSS(const FileName: TRVUnicodeString[1099];
  AOptions: TRVSaveCSSOptions;
  Encoding: TRVHTMLEncoding[1078]): Boolean;
```

(changed in versions 18 and 21)

Use this method if you want to save content of TRichView as HTML file with external CSS, see TRichView[216].HTMLSaveProperties[244].ExternalCSSFileName[440].

You can save CSS only once and use it for several HTML files (if they are linked to the same or compatible TRVStyle components) .

This method allows reducing size of HTML files.

The following options may be included in the **Options** parameter.

Option	Meaning
rvcssNoDefCSSStyle	Corresponds to *rvcsssoNoDefCSSStyle* in TRichView.HTMLProperties.CSSOptions[438]. It's not recommended to use this option.
rvcssDefault0Style	Properties of TextStyles[764][0] are not saved. Corresponds to TRichView.HTMLProperties.Default0Style[439].
rvcssMarkersAsText	Lists will be saved as text. TRichView.HTMLProperties.ListMarkerSavingType[444] = *rvhtmlmstAsText*.

Please do not use other options.

There are no methods for loading CSS files, CSS files are loaded and parsed automatically when loading HTML files.

See also:

- TRichView working with HTML files[132].

6.2.7.2.15 TRVStyle.SaveCSSToStream

The same as SaveCSS[772], but accepts **Stream** parameter

```
procedure SaveCSSToStream(Stream: TStream; Options: TRVSaveCSSOptions);
```

(introduced in v1.1.4)

6.2.7.2.16 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[771];
- SaveReg[773];
- SaveCSS[772].

6.2.7.2.17 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[773].

See also:

- LoadReg[771];
- SaveINI[773].

6.2.7.3 Events

In TRVStyle

- OnAfterApplyStyle[774]
- OnApplyStyle[774]
- OnApplyStyleColor[775]
- OnDrawCheckpoint[776]
- OnDrawPageBreak[778]
- OnDrawParaBack[780]
- OnDrawStyleText[781]
- OnDrawTextBack[783]
- OnStyleHoverSensitive[784]

6.2.7.3.1 TRVStyle.OnAfterApplyStyle

Occurs after RichView applies attributes of the **StyleNo**-th style to Canvas.

```
property OnAfterApplyStyle: TRVAfterApplyStyleEvent;
```

VCL and LCL:

```
type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      StyleNo: Integer) of object;
```

(introduced in version 12)

FireMonkey:

```
type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; Canvas: TRVFMXCanvas[1018];
      StyleNo: Integer) of object;
```

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[774], but occurs after the default procedure. This event does not occur if OnApplyStyle[774] returned *False* in DoDefault parameter.

Input parameters:

Sender – TRVStyle generating this event;

Canvas – canvas for applying font attributes to.

See also:

- OnApplyStyleColor[775];
- OnDrawStyleText[781].

See also properties:

- TextStyles[764].

6.2.7.3.2 TRVStyle.OnApplyStyle

Occurs when RichView applies attributes of the **StyleNo**-th style to Canvas.

```
property OnApplyStyle: TRVApplyStyleEvent;
```

VCL and LCL:

```
type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; ACanvas: TCanvas;
      StyleNo: Integer; PSaD: PRVScreenAndDevice[1090];
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3; modified in v18)

FireMonkey:

```
type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas[1018];
```

```
      StyleNo: Integer; PSaD: PRVScreenAndDevice[1090];
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3; modified in v18)

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.

PSaD – pointer to a record used for conversion between TRVStyleLength[1094] values and device pixels. It may be *nil* when applying to the screen.

DoDefault is equal to True.

Output parameters:

DoDefault – set to *False* to prevent the default style applying procedure and OnAfterApplyStyle[774] event.

See also:

- OnApplyStyleColor[775];
- OnDrawStyleText[781].

See also properties:

- TextStyles[764].

6.2.7.3.3 TRVStyle.OnApplyStyleColor

Occurs when RichView applies the color of the **StyleNo**-th style to **Canvas**.

```
property OnApplyStyleColor: TRVApplyStyleColorEvent;
```

VCL and LCL:

```
type
  TRVApplyStyleColorEvent =
    procedure (Sender: TRVStyle; ACanvas: TCanvas;
      StyleNo: Integer; DrawState: TRVTextDrawStates[1097];
      ApplyTo: TRVColoredParts;
      Printing: Boolean; ColorMode: TRVColorMode[1063];
      DarkMode: Boolean;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3, changed in version 23)

FireMonkey:

```
type
  TRVApplyStyleColorEvent =
    procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas[1018];
      StyleNo: Integer; DrawState: TRVTextDrawStates[1097];
      ApplyTo: TRVColoredParts;
      Printing: Boolean; ColorMode: TRVColorMode[1063];
      DarkMode: Boolean;
```

```
    var DoDefault: Boolean) of object;
```

where

```
  TRVColoredPart = (rvcpBack, rvcpText);
  TRVColoredParts = set of TRVColoredPart;
```

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 [1097].

ApplyTo identifies object which require coloring.

Printing – if True, colors are requested for printing; otherwise, for drawing on the screen.

ColorMode – color mode.

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData [1014] before using.

Sender.**ItemNo**: Integer, index of the item in **Sender**.**RVData**. Can be -1 if drawing is not for an item (for example, for a paragraph mark)

Output parameters:

DoDefault – set to *False* to prevent the default colors applying procedure.

FireMonkey note: to apply text color, use **Sender**.GraphicInterface.SetTextColor(Canvas, Color) method.

See also:

- OnApplyStyle [774];
- OnDrawStyleText [781].

See also properties:

- TextStyles [764].

6.2.7.3.4 TRVStyle.OnDrawCheckpoint

Occurs when drawing *checkpoint* [92].

```
property OnDrawCheckpoint: TRVDrawCheckpointEvent;
```

VCL and LCL:

```
type
  TRVDrawCheckpointEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      X,Y: TRVCoord[1064]; ItemNo: Integer; XShift: TRVCoord[1064];
```

```
      RaiseEvent: Boolean; Control: TControl;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

FireMonkey:

```
type
  TRVDrawCheckpointEvent =
    procedure (Sender: TRVStyle; Canvas: TRVFMXCanvas[1018];
      X,Y: TRVCoord[1064]; ItemNo: Integer; XShift: TRVCoord[1064];
      RaiseEvent: Boolean; Control: TControl;
      var DoDefault: Boolean) of object;
```

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 [247].

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 [1014] 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 [743], 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[1055];
  CheckpointTag: TRVTag[1096];
  CheckpointName: String;
```

```
begin
  if ItemNo>=0 then
  begin
    CheckpointData :=
      TCustomRVFormattedData[1014](Sender.RVData).GetItemCheckpoint[305](
        ItemNo));
    TCustomRVFormattedData(Sender.RVData).GetCheckpointInfo[299](
      CheckpointTag, CheckpointName, RaiseEvent);
    ...
  end;
end;
```

See also properties:

- CheckpointColor, CheckpointEvColor[743].

See also:

- Checkpoints overview[92].

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.7.3.5 TRVStyle.OnDrawPageBreak

Occurs when drawing page breaks.

```
property OnDrawPageBreak: TRVDrawPageBreakEvent;
```

VCL and LCL:

```
type
  TRVPageBreakType = (rvpbSoftPageBreak, rvpbPageBreak);

  TRVDrawPageBreakEvent =
    procedure (Sender: TRVStyle;
      ACanvas: TCanvas;
      Y, XShift: TRVCoord[1064];
      PageBreakType: TRVPageBreakType;
      Control: TControl;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3; modified in 1.7)

FireMonkey:

```
type
  TRVPageBreakType = (rvpbSoftPageBreak, rvpbPageBreak);

  TRVDrawPageBreakEvent =
    procedure (Sender: TRVStyle;
      ACanvas: TRVFMXCanvas[1018];
      Y, XShift: TRVCoord[1064];
      PageBreakType: TRVPageBreakType;
      Control: TControl;
      var DoDefault: Boolean) of object;
```

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[247].

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[1014] 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[901] 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[756] or SoftPageBreakColor[761], depending on **PageBreakType**.

Using **Sender**.**RVData** you can get additional information about the item:

For example:

```
uses ..., CRVFData;
...
var ParaNo: Integer;
begin
  ParaNo := TCustomRVFormattedData[1014](Sender.RVData).GetItemPara[309](
    Sender.ItemNo));
  ...
end;
```

See also properties:

- PageBreakColor[756];
- SoftPageBreakColor[761].

See also properties of TCustomRichView[216]:

- PageBreaksBeforeItem[250].

See also methods of TCustomRichView[216]:

- AssignSoftPageBreaks[284].

See also methods of TCustomRichViewEdit[469]:

- InsertPageBreak[534];
- RemoveCurrentPageBreak[551].

See also properties of table[901]:

- PrintOptions[919].

See also properties of table row[966]:

- PageBreakBefore[967].

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.7.3.6 TRVStyle.OnDrawParaBack

Occurs when painting a background of paragraph of the **ParaNo**-th style.

```
property OnDrawParaBack: TRVDrawParaRectEvent;
```

VCL and LCL:

```
type
  TRVDrawParaRectEvent =
    procedure (Sender: TRVStyle; ACanvas: TCanvas;
      ParaNo: Integer; ARect: TRVCoordRect[1064];
      var DoDefault: Boolean) of object;
```

FireMonkey:

```
type
  TRVDrawParaRectEvent =
    procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas[1018];
      ParaNo: Integer; ARect: TRVCoordRect[1064];
      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[757]).

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData[1014] 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[757];
- TParaInfo[828].Background[830].

See also:

- RichView Paragraphs[100].

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.7.3.7 TRVStyle.OnDrawStyleText

Allows to perform custom drawing of a text item of the **StyleNo**-th text style.

```
property OnDrawStyleText: TRVDrawStyleTextEvent
```

VCL and LCL:

```
type
  TRVDrawStyleTextEvent = procedure (Sender: TRVStyle[739];
    const s: TRVUnicodeString[1099]; ACanvas: TCanvas; StyleNo: Integer;
    SpaceAtLeft, ALeft, ATop, AWidth, AHeight
    {$IFDEF RVUSEBASELINE},BaseLine{$ELSE}
    {$IFDEF RVHIGHFONTS},DrawOffsY{$ENDIF}
    {$ENDIF}: TRVCoord[1064];
    TextWidth: TRVCoord[1064]; SpaceCount: Integer;
    DrawState: TRVTextDrawStates[1097];
    Printing, PreviewCorrection, ForMetafile: Boolean;
    ColorMode: TRVColorMode[1063]; DarkMode: Boolean;
    DefBiDiMode: TRVBiDiMode[1058];
    RefCanvas: TCanvas;
    LeftNoExpIndex, RightNoExpIndex: Integer;
    PSaD: PRVScreenAndDevice[1090];
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3; new states in TRVTextDrawStates[1097] since 1.4; changed in v16, v18, v23)

FireMonkey:

```
type
  TRVDrawStyleTextEvent = procedure (Sender: TRVStyle[739];
    const s: TRVUnicodeString[1099]; ACanvas: TRVFMXCanvas[1018];
    StyleNo: Integer;
    SpaceAtLeft, ALeft, ATop, AWidth, AHeight
    {$IFDEF RVUSEBASELINE},BaseLine{$ELSE}
    {$IFDEF RVHIGHFONTS},DrawOffsY{$ENDIF}
    {$ENDIF}: TRVCoord[1064];
    TextWidth: TRVCoord[1064]; SpaceCount: Integer;
    DrawState: TRVTextDrawStates[1097];
    Printing, PreviewCorrection, ForMetafile: Boolean;
    ColorMode: TRVColorMode[1063]; DarkMode: Boolean;
```

```
    DefBiDiMode: TRVBiDiMode[1058];
    RefCanvas: TCanvas;
    LeftNoExpIndex, RightNoExpIndex: Integer;
    PSaD: PRVScreenAndDevice[1090];
    var DoDefault: Boolean) of object;
```

You can use this event to modify drawing of text of some or all text styles.

Input parameters:

The same is in TFontInfo[821].Draw[825]. You can call **Sender**.TextStyles[764][**StyleNo**].Draw(..) passing the even parameters.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData[1014] 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[1096];
...
ItemTag := TCustomRVFormattedData[1014](Sender.RVData).GetItemTag[310](
  Sender.ItemNo));
```

Output parameters:

DoDefault – set to *False* to prevent default drawing.

Note: In VCL/LCL, RVUSEBASELINE is defined by default, but you can disable this definition in RV_Defs.inc.
In FireMonkey, RVUSEBASELINE is not defined, and FireMonkey version does not support drawing relative to a base line.
RVHIGHFONTS is defined in all frameworks by default.

See also events:

- OnApplyStyle[774];
- OnApplyStyleColor[775];
- OnDrawTextBack[783].

See also properties:

- TextStyles[764].

Demo projects:

- Demos*\Assorted\Custom Draw\

6.2.7.3.8 TRVStyle.OnDrawTextBack

Occurs when drawing background of text of the **StyleNo**-th text style.

```
property OnDrawTextBack: TRVDrawTextBackEvent;
```

VCL/LCL:

```
type
  TRVDrawTextBackEvent =
    procedure (Sender: TRVStyle; ACanvas: TCanvas;
      StyleNo: Integer; ALeft, ATop, AWidth, AHeight: TRVCoord[1064];
      DrawState: TRVTextDrawStates[1097];
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

FireMonkey:

```
type
  TRVDrawTextBackEvent =
    procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas[1018];
      StyleNo: Integer; ALeft, ATop, AWidth, AHeight: TRVCoord[1064];
      DrawState: TRVTextDrawStates[1097];
      var DoDefault: Boolean) of object;
```

You can use this event to modify drawing of text background for some or all text styles.

Unlike OnDrawStyleText[781], this event is called only once for each drawing item (OnDrawStyleText[781] can be called several times if part of text item is selected). In paragraphs having Alignment[829]=*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[764] collection.

Left, **Top**, **Width**, **Height** – rectangle where to paint.

DrawState may include *rvtsControlFocused* and *rvtsHover*, see , see TRVTextDrawStates[1097].

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData[1014] 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[1096];
...
```

```
ItemTag := TCustomRVFormattedData(Sender.RVData).GetItemTag(
  Sender.ItemNo));
```

Output parameters:

DoDefault – set to *False* to prevent default drawing. It is supported only if TextBackgroundKind <>*rvtbkSimple*.

See also events:

- OnDrawStyleText
- OnApplyStyle;
- OnApplyStyleColor;

See also properties:

- TextStyles.

Demo projects:

- Demos*\Assorted\Custom Draw\

6.2.7.3.9 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 styles.

Input parameters:

Sender – TRVStyle generating the event.

StyleNo – index of text style in TextStyles collection.

Sensitive – *True* if for the given text style (HoverEffects =[]) or (HoverBackColor <>BackColor) or (HoverColor <>*clNone*) or (HoverUnderlineColor <>*clNone*), or if HoverColor <>*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;
- OnDrawTextBack.

See also properties:

- TextStyles [764].

6.2.7.4 Classes of properties - Style collections

Classes defining formatting

- TFontInfos [785] – collection of items representing character formatting, a type of TRVStyle [739] .TextStyles [764] property
- TParaInfos [791] – collection of items representing paragraph formatting, a type of TRVStyle.ParaStyles [757] property
- TRVListInfos [795] – collection of items representing paragraph lists (bullets and numbering), a type of TRVStyle.ListStyles [755] property

"Real styles"

- TRVStyleTemplateCollection [797] – collection of items representing named styles, a type of TRVStyle.StyleTemplates [762] property

6.2.7.4.1 TFontInfos

Collection of text attributes (styles). Class of TRVStyle.TextStyles [764] property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
TFontInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items [786] – array of TFontInfo [821].

InvalidItem [786] is returned when accessing item which is out of index range.

Methods for Text Style Search

- FindStyleWithFontStyle [790]
- FindStyleWithFontSize [790]
- FindStyleWithColor [788]
- FindStyleWithFontName [789]
- FindSuchStyle, FindSuchStyleEx [791]
- FindStyleWithFont [789]
- FindStyleWithCharset [788]

See Also

- TParaInfos [791] – class of collection of paragraph attributes (styles);
- TRVListInfos [795] – class of collection of list attributes (i.e. styles of bullets and numbering).

6.2.7.4.1.1 Properties

In TFontInfos

InvalidItem [786]
Items [786]

Derived from TCollection

Count

This item is returned when accessing Items [786][ItemNo], if (ItemNo<0) or (ItemNo>=TFontInfos.Count)

```
property InvalidItem: TFontInfo [821];
```

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value:

Default value has all properties Items [786][0], but red background [808] and white text color [810].

See also:

- TParaInfos.InvalidItem [792].

Provides indexed access to the items in the collection of text attributes (styles)

```
property Items[Index:Integer]: TFontInfo [821];
```

This is the default property of TFontInfos, so you can write

```
MyRVStyle.TextStyles [764][i]
```

instead of

```
MyRVStyle.TextStyles.Items[i]
```

Many methods of components inherited from TCustomRichView [216] have StyleNo parameter, which is an index in TFontInfos.Items collection (more specifically, in TCustomRichView.Style [260].TextStyles [764].Items).

Example:

```
  MyRichView.AddNL [277]('Hello', 0, 0);
```

Unlike normal collections, when accessing item with invalid Index (for example, Index<0 or Index>=Count), no exception occurs. InvalidItem [786] is returned instead.

Indicates the number of pixels that make up a logical inch in the vertical direction.

```
PixelsPerInch: Integer;
```

DEPRECATED!

Use TRichView[216].DocumentPixelsPerInch[239] instead of this property.

This property affects how text is displayed in components inherited from TCustomRichView[216] and TRVReportHelper[721] (this property is ignored by TRVPrint[696]).

Note: this property is not available in FireMonkey version

Default value:

0

6.2.7.4.1.2 Methods

In TFontInfos

Add[787]
Create[788]
FindStyleWithCharset[788]
FindStyleWithColor[788]
FindStyleWithFont[789]
FindStyleWithFontName[789]
FindStyleWithFontSize[790]
FindStyleWithFontStyle[790]
FindSuchStyle[791]
FindSuchStyleEx[791]

Derived from TCustomRVInfos

AssignTo[788]

You can also use methods derived from TCollection

Adds a new style in the collection of text styles

```
function Add: TFontInfo[821];
```

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[786] array.

Return value:

Created collection item.

See also:

- TFontInfo.Create[825].

Adds a new item in the collection of text styles, initializes it.

```
function AddFont(Name: TFontName; Size: Integer;
  Color, BackColor: TRVColor[1063]; Style:TFontStyles): TFontInfo[821];
```

Please use Add[787] instead of this method.

Adds a new item in the collection of text styles, initializes it.

```
function AddFontEx(Name: TFontName; Size: Integer;
  Color, BackColor: TRVColor[1063]; Style:TFontStyles;
  Charset: TRVFontCharset[1075]): TFontInfo[821];
```

Please use Add[787] 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[805]) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.TextStyles[764]);
```

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[764] property.

Searches for style with the specified **Charset**.

```
function FindStyleWithCharset(BaseStyle: Integer;
  Charset: TRVFontCharset[1075]): Integer;
```

(Introduced in version 1.4)

Searches for style having all properties of the **BaseStyle**-th style, but Charset[809]=**Charset**.

BaseStyle is an index in Items[786].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

Searches for style with the specified colors.

```
function FindStyleWithColor(BaseStyle: Integer;
  Color, BackColor: TRVColor[1063]): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but Color[810]=**Color** and BackColor[808]=**BackColor**.

BaseStyle is an index in Items[786].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

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[786].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

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[811]=**FontName**.

BaseStyle is an index in Items[786].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

Searches for style having the specified font size.

```
function FindStyleWithFontSize(BaseStyle: Integer;
  Size: TRVFontSize[1076]): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but font Size[817]=**Size**.

This method does not return items having odd values of SizeDouble[817].

BaseStyle is an index in Items[786].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

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[786].

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[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** methods; the most universal method is FindSuchStyle[791];
- TFontInfo.IsEqual[821].

Searches for a style.

```
function FindSuchStyle(BaseStyle: Integer; Style: TFontInfo[821];
  Mask: TRVFontInfoProperties[1075]): Integer;
function FindSuchStyleEx(BaseStyle: Integer; Style: TFontInfo[821];
  Mask: TRVFontInfoProperties[1075]): 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[786].

Mask is a set of properties to compare (other properties are ignored). Use RVAllFontInfoProperties[1075] constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*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[804]=*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[771];
- TFontInfo.IsEqual[821];
- TParaInfos.FindSuchStyle[794];
- TRVListInfos.FindSuchStyle[796].

Demos:

- Demos*\Editors\Editor 2\

(see procedure TForm1.rveStyleConversion, rve.OnStyleConversion[595] event)

6.2.7.4.2 TParaInfos

Collection of paragraph attributes (styles). Class of TRVStyle.ParaStyles[757] property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
TParaInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items [792] – array of TParaInfo [828].

InvalidItem [792] is returned when accessing item which is out of index range.

Methods for Paragraph Style Search

- FindStyleWithAlignment [794];
- FindSuchStyle, FindSuchStyleEx [794].

See Also

- TFontInfos [785] – class of collection of text attributes (styles);
- TRVListInfos [795] – class of collection of list attributes (i.e. styles of bullets and numbering).

6.2.7.4.2.1 Properties

In TParaInfos

InvalidItem [792]
Items [792]

Derived from TCollection

Count

This item is returned when accessing Items [792][ItemNo], if (ItemNo<0) or (ItemNo>=TParaInfos.Count)

```
property InvalidItem: TParaInfo [828];
```

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value

Default value has all properties Items [792][0], but (LeftIndent [832],SpaceBefore [836],RightIndent [835] ,SpaceAfter [835])=(1,1,1,1) and red border [831].

See also

- TFontInfos.InvalidItem [786].

Provides indexed access to the items in the collection of paragraph attributes (styles)

```
property Items[Index: Integer]: TParaInfo [828];
```

This is default property of TParaInfos, so you can write

```
MyRVStyle.ParaStyles [757][i]
```

instead of

```
MyRVStyle.ParaStyles.Items[i]
```

Many methods of components inherited from TCustomRichView [216] have ParaNo parameter, which is an index in TParaInfos.Items collection (more specifically, in TCustomRichView.Style [260] .ParaStyles [757].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.

6.2.7.4.2.2 Methods

In TParaInfos

Add
Create
FindStyleWithAlignment
FindSuchStyle
FindSuchStyleEx

Derived from TCustomRVInfos

AssignTo

You can also use methods derived from TCollection

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[757] 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[829]=**Alignment**.

BaseStyle is an index in Items[792].

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style or -1, if not found.

See also methods:

- FindSuchStyle[794].

Searches for a style having properties like **Style**.

```
function FindSuchStyle(BaseStyle: Integer; Style: TParaInfo[828];
  Mask: TRVParaInfoProperties[1083]): Integer;
function FindSuchStyleEx(BaseStyle: Integer; Style: TParaInfo[828];
  Mask: TRVParaInfoProperties[1083]): Integer;
```

(Introduced in version 1.5 and version 14)

The **BaseStyle**-th style is checked first. **BaseStyle** is an index in Items[792].

Mask is a set of properties to compare (other properties are ignored). Use RVAllParaInfoProperties[1083] constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*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[804]=*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[794];
- TRVStyle.FindParaStyle[771];
- TParaInfo.IsEqual[837];

- TFontInfos.FindSuchStyle [791];
- TRVListInfos.FindSuchStyle [796].

Demos:

- Demos*\Editors\Editor 2\

(see procedure TForm1.rveParaStyleConversion, rve.OnParaStyleConversion [592] event)

6.2.7.4.3 TRVListInfos

Collection of paragraph list attibutes (bullets and numbering). Class of TRVStyle.ListStyles [755] property.

Unit RVStyle.

Syntax

```
TCustomRVInfos = class(TCollection)
TRVListInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items [795] – array of TRVListInfo [841].

Main Methods

- FindSuchStyle [796]
- FindStyleWithLevels [796]

See Also

- TFontInfos [785] – class of collection of text attributes (styles);
- TParaInfos [791] – class of collection of paragraph attributes (styles);
- List markers item type [180].

6.2.7.4.3.1 Properties

In TRVListInfos

Items [795]

Derived from TCollection

Count

Provides indexed access to items in the collection of paragraph list attributes (bullets and numbering)

```
property Items[Index:Integer] : TRVListInfo[841];
```

This is default property of TRVListInfos, so you can write

```
MyRVStyle.ListStyles[755][i]
```

instead of

```
MyRVStyle.ListStyles.Items[i]
```

6.2.7.4.3.2 Methods

In TRVListInfos

Add[797]
FindStyleWithLevels[796]
FindSuchStyle[796]

Derived from TCustomRVInfos

AssignTo[797]

You can also use methods derived from TCollection

Searches for the style. Adds if not found (optionally)

```
function FindSuchStyle(Style: TRVListInfo[841];
  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[804] property = *False*.

The following properties are ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

Return value:

Index of style, or -1 if not found (and not added).

See also:

- FindStyleWithLevels[796];
- TFontInfos.FindSuchStyle[791];
- TParaInfos.FindSuchStyle[794].

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[842] and **StyleNameForAdding** to its StyleName[805]. The created style has Standard[804] property = *False*.

Return value:

Index of style, or -1 if not found (and not added).

See also:

- FindSuchStyle[796].

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[805]) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.ListStyles[755]);
```

Adds a new style to the collection of paragraph list styles (bullets and numbering)

```
function Add: TRVListInfo[841];
```

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[795] array.

Add returns the new collection item.

6.2.7.4.4 TRVStyleTemplateCollection

Collection of "real" styles of text and paragraphs. Class of TRVStyle.StyleTemplates[762] property.

Unit RVStyle.

Syntax

```
TRVStyleTemplateCollection = class(TCollection)
```

(introduced in version 14)

Hierarchy

TObject
TPersistent
TCollection

Main Properties

- Items[798] – array of TRVStyleTemplate[843].
- DefStyleName[798] – a format string used to generate names of new items.

Main Methods

- AssignStyleTemplates[799] assigns another collection to this collection;
- FindById, FindItemById[800] search for items by their identifiers;
- FindByName, FindItemByName[800] search for items by their names;
- LoadFromRVST, SaveToRVST[801] save and load the collection to a file;
- LoadFromStreamRVST, SaveToStreanRVST[802] save and load the collection to a stream (Delphi 10.3+);
- ResetNameCounter[802] resets a counter used to generate names for new items.

See also

- Styles and style templates [103].

6.2.7.4.4.1 Properties

In TRVStyleTemplateCollection

- DefStyleName [798]
- ForbiddenIds [798]
- Items [798]
- NormalStyleTemplate [799]

Derived from TCollection

- Count

Specifies the name for new style templates.

```
property DefStyleName: TRVUnicodeString[1099];
```

(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 [845] of new items in this collection.

Default value:

'Style %d'

See also:

- ResetNameCounter [802]

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 [508]. When the user deletes a style template, add its Id [844] 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[843];
```

This is default property of TRVStyleTemplateCollection, so you can write

```
MyRVStyle.StyleTemplates[762][i]
```

instead of

```
MyRVStyle.StyleTemplates.Items[i]
```

Returns the item having Name[845]='Normal'.

```
property NormalStyleTemplate: TRVStyleTemplate[843];
```

If such an item does not exist in Items[798], the property returns *nil*.

6.2.7.4.4.2 Methods

In TRVStyleTemplateCollection

AssignStyleTemplates[799]
AssignToStrings[799]
ClearParaFormat[800]
ClearTextFormat[800]
FindById[800]
FindByName[800]
FindItemById[800]
FindItemByName[800]
LoadFromRVST[801]
LoadFromStreamRVST[802] (D10.3+)
ResetNameCounter[802]
SaveToRVST[801]
SaveToStreamRVST[802] (D10.3+)

Copies items from **Source** to this collection.

```
procedure AssignStyleTemplates(Source: TRVStyleTemplateCollection;
  CopyIds: Boolean);
```

If **CopyIds**=*True*, Id[844] properties of **Source**.Items[798] are copied to new items.

If **CopyIds**=*False*, the method generates new Id for new items, retaining NextId[846] and ParentId[847] relations.

Note:

StyleTemplates1.Assign(StyleTemplates2) is equivalent to StyleTemplates1.AssignStyleTemplates(StyleTemplates2, False).

See also:

- TCustomRichViewEdit.ChangeStyleTemplates[508].

Assign style templates to a string list.

```
type
  TRVStyleTemplateKinds = set of TRVStyleTemplateKind[845];
procedure AssignToStrings(Strings: TStrings; AssignIds: Boolean;
  Kinds: TRVStyleTemplateKinds; ExcludeThisStyle: TRVStyleTemplate[843];
  OnlyPotentialParents: Boolean);
```

Parameters:

Strings – a destination string list. Names[845] of style templates are added as items to this string list.

If **AssignIds**=*True*, the method saves Ids [844] of style templates as Objects of **Strings** (type-casted to TObject).

Only style templates having Kinds [845] 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 [845] 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[806]);
procedure ClearParaFormat(AParaStyle: TCustomRVParaInfo[828]);
```

ClearTextFormat clears formatting from **ATextStyle**. The method resets all its properties to default values, then:

- if **ATextStyle** is TFontInfo [821], and **ATextStyle**.Jump [823]=True, and "Hyperlink" style template exists in Items [798], 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 [798], the method applies it to **AParaStyle**;
- otherwise, the method assigns **AParaStyle**.StyleTemplateId = -1.

Searches for the item having the specified identifier [844] **Id**.

```
function FindById(Id: TRVStyleTemplateId): Integer;
function FindItemById(Id: TRVStyleTemplateId): TRVStyleTemplate;
```

FindById returns the index of the found item in Items [798], or -1 if not found.

FindItemById returns the the found item, or *nil* if not found.

See also:

- FindByName, FindItemByName [800]

Searches for the item having the specified name [845] **Name**

```
function FindByName(const Name: TRVStyleTemplateName[1094]): Integer;
function FindItemByName(const Name: TRVStyleTemplateName[1094]):
  TRVStyleTemplate;
```

FindByName returns the index of the found item in Items [798], or -1 if not found.

FindItemByName returns the the found item, or nil if not found.

See also:

- FindById, FindItemById [800]

Applies styles from the file to the collection.

```
function InsertFromRVST(const FileName: TRVUnicodeString[1099];
  Units: TRVStyleUnits[1094]): Boolean;
```

Files of "RichView Styles" format contain style templates. Normally they have "rvst" extension. They can be created by SaveToRVST[802] 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[797];
begin
  if not OpenDialog1.Execute then
    exit;
  StyleTemplates := TRVStyleTemplateCollection[797].Create(nil);
  try
    StyleTemplates.AssignStyleTemplates[799](rvs.StyleTemplates, True);
    if not StyleTemplates.InsertFromRVST(OpenDialog1.FileName,
      RichViewEdit1.Style[260].Units[764]) then
      exit;
    RichViewEdit1.ChangeStyleTemplates[508](StyleTemplates);
  finally
    StyleTemplates.Free;
  end;
end;
```

The methods load and save style templates to files.

```
function LoadFromRVST(const FileName: TRVUnicodeString[1099];
  Units: TRVStyleUnits[1094];
  ARVStyle: TRVStyle[739] = nil): Boolean;
function SaveToRVST(const FileName: TRVUnicodeString[1099];
  Units: TRVStyleUnits[1094]): 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[762], this is RVStyle.Units[764];
- 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**.

ARVStyle parameter is necessary if you load styles into the collection that is not linked to any TRVStyle[739] component. Specify TRVStyle component which will be used for unit conversion (if the file is measured in units different from **Units**).

Note: after loading to RVStyle.StyleTemplates[762], clear RVStyle.TextStyles[764] and ParaStyles[757] (because the old items may contain references[805] to previous style templates). Then you can add default text and paragraph styles (see the example[848]).

Example:

Demos*\Editors\StyleTemplates\ demo loads "default.rvst" in "File | New" command.

See also

- InsertFromRVST[801]
- LoadFromStreamRVST, SaveToStreamRVST[802]

The methods load and save style templates to streams.

```
function SaveToStreamRVST(Stream: TStream;
  Units: TRVStyleUnits[1094]): Boolean;
function LoadFromStreamRVST(Stream: TStream;
  Units: TRVStyleUnits[1094]; ARVStyle: TRVStyle[739] = nil): Boolean;
```

(introduced in version 19; for Delphi and C++Builder 10.3+)

These methods are analogs SaveToRVST and LoadFromRVST[801], but they save/load to/from **Stream** instead of a file.

Resets the counter used to generate unique names for new Items[798].

```
procedure ResetNameCounter;
```

Names are generated using DefStyleName[798] 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.

6.2.7.5 Classes of properties - Styles

Classes defining formatting

Main classes:

- TFontInfo[821] (inherited from TCustomRVFontInfo[806]) – a type of items in TRVStyle[739].TextStyles[764] property collection, contains character properties.
- TParaInfo[837] (inherited from TCustomRVParaInfo[828]) – a type of items in TRVStyle.ParaStyles[757] property collection, contains paragraph properties.
- TRVListInfo[841] – a type of items in TRVStyle.ListStyles[755] property collection, contains properties of paragraph lists (bullets and numbering)

Ancestor classes:

- TCustomRVInfo, TCustomRVTextOrParaInfo[803]

"Real styles"

- TRVStyleTemplate [843] – a type of items in TRVStyle.StyleTemplates [762] property, represent a named style.

6.2.7.5.1 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

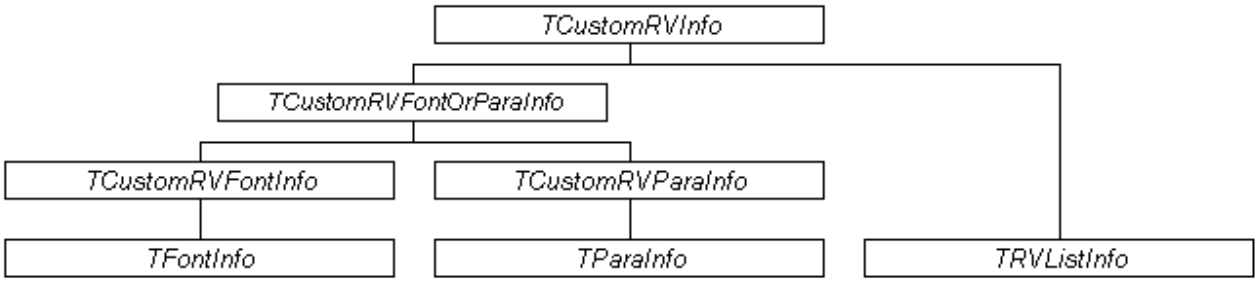
Using

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 [806]	This class introduces the most of properties related to text attributes. Ancestor of TFontInfo.
TCustomRVParaInfo [828]	This class introduces the most of properties related to paragraph attributes. Ancestor of TParaInfo.
TFontInfo [821]	Text style, defines text attributes. Class of items in the collection TRVStyle [739].TextStyles [764].
TParaInfo [837]	Paragraph style, defines paragraph attributes. Class of items in the collection TRVStyle [739].ParaStyles [757].
TRVListInfo [841]	List style (i.e. style of paragraphs' bullets & numbering). It has Levels [842] collection defining attributes of paragraph list markers [180]. Class of items in the collection TRVStyle [739] .ListStyles [755].

The class hierarchy is explained on the diagram below.



Properties

TCustomRVInfo introduces the following properties:

- Standard [804];
- StyleName [805].

TCustomRVFontOrParaInfo introduces the following properties:

- StyleTemplateId [805].

6.2.7.5.1.1 Properties

In TCustomRVFontOrParaInfo

- StyleTemplateId [805]

In TCustomRVInfo

- Standard [804]
- StyleName [805]

Identifies standard (essential) styles

```
property Standard: Boolean;
```

(introduced in version 1.6)

Description

Standard styles are not deleted by DeleteUnusedStyles [293].

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 [771], TRVStyle.FindParaStyle [771], TFontInfos.FindSuchStyleEx [791], TParaInfos.FindSuchStyleEx [794] are not standard.

Styles which are added automatically while loading from RVF preserve value of this property.

Usage without StyleTemplates [762]

If StyleTemplates are not used [263], 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 [224], 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 [762]

If StyleTemplates are used, it's recommended to have only one standard paragraph style initially, linked [805] to NormalStyleTemplate [799], and only one standard text style not linked to any style template (but its ParaStyleTemplateId [815] should point to NormalStyleTemplate [799] as well). All other styles will be added on editing.

Default value:

True

See also:

- TFontInfo [821] – class of text attributes (style);
- TParaInfo [828] – class of paragraph attributes (style);
- TRVListInfo [841] – class of paragraph list attributes (style);
- RichViewResetStandardFlag [1115] typed constant.

Name of this style.

```
property StyleName: TRVUnicodeString[1099];
```

(changed in version 18)

This name is used:

- when displaying a design-time editor for collections of styles;
- when assigning text styles to TStrings [788].
- when saving RTF without using StyleTemplates [263] and with *rvrtfSaveStyleSheet* [252] option;

This property is ignored when comparing and searching styles (see also RichViewCompareStyleNames [1111] typed constant).

Names of styles do not need to be unique in the collection (unless you save RVF with *rvfoUseStyleNames* [254] option).

This property makes sense if you use a predefined set of styles [224]. If you add styles dynamically, you can ignore this property.

Links this style to a style template.

```
property StyleTemplateId: TRVStyleTemplateId[1094];
```

(introduced in version 14)

This property is used only if style templates are used [263].

TCustomRVFontInfo [806] may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkText*.

TCustomRVParaInfo[828] may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkPara*.

See also:

- TCustomRVFontInfo.ParaStyleTemplateId[815]

6.2.7.5.2 TCustomRVFontInfo

This is an ancestor class for TFontInfo[821] (class of item in collection of text styles[764]).

TCustomRVFontInfo introduces the most of properties of text.

Unit RVStyle.

Syntax

```
TCustomRVFontInfo = class(TCustomRVFontOrParaInfo[803])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVFontOrParaInfo[803]
TCustomRVInfo[803]

Properties

Font attributes (similar to properties of TFont):

- FontName[811] – font name;
- Size[817] and SizeDouble[817] – font size;
- Color[810] – text color;
- Style[817] – text styles (bold, italic, underline, strikeout);
- Charset[809] – font character set (for non-Unicode text styles).

Additional properties affecting text appearance and layout:

- BackColor[808] – text background color;
- UnderlineColor[818] – underline color;
- UnderlineType[819] – underline style (normal, double, dotted, etc.);
- StyleEx[818] – additional text styles (overline, all-caps);
- SubSuperScriptType[818] – normal text/subscript/superscript;
- VShift[820] moves text up or down from its base line;
- CharScale[808] – horizontal character scaling;
- CharSpacing[809] – spacing between characters;
- BiDiMode[808] – default text direction.
- EmptyWidth[810] – width for empty text item

Hypertext properties:

- HoverColor[812] – text highlight color;
- HoverBackColor[811] – text background highlight color;
- HoverUnderlineColor[813] – underline highlight color;
- HoverEffects[812] – additional highlight effects (underline);
- JumpCursor[813] – mouse cursor for this text.

Properties related to editing in TCustomRichViewEdit[469]:

- Protection[815] – text protection options;
- EmptyWidth[810] – width for empty text item, affects editing.

Properties related to text export:

- Options[814] allows to create RTF and HTML codes.

Methods

IsEqual[821] compares this text style with another one.

You can use Assign[820] method to assign:

- TCustomRVFontInfo to TCustomRVFontInfo;
- TFont to TCustomRVFontInfo;
- TCustomRVFontInfo to TFont.

See Also

Classes:

- TFontInfo[821] – class inherited from this class, text attributes (style);
- TFontInfos[785] – collection of text attributes (styles) (TFontInfo[821]).

6.2.7.5.2.1 Properties

In TCustomRVFontInfo

- BackColor[808]
- BiDiMode[808]
- CharScale[808]
- Charset[809]
- CharSpacing[809]
- Color[810]
- EmptyWidth[810]
- FontName[811]
- HoverBackColor[811]
- HoverColor[812]
- HoverEffects[812]
- HoverUnderlineColor[813]
- JumpCursor[813]
- Options[814]
- ParaStyleTemplateId[815]
- Protection[815]
- Size[817]
- SizeDouble[817]
- Style[817]
- StyleEx[818]
- SubSuperScriptType[818]
- UnderlineColor[818]
- UnderlineType[819]
- VShift[820]

Derived from TCustomRVFontOrParaInfo [803]

- StyleTemplateId [805]

Derived from TCustomRVInfo [803]

- Standard [804]
- StyleName [805]

Background color for text of this style

```
property BackColor: TRVColor [1063];
```

Set BackColor=*rvclNone* [1105] for transparent text background.

Default value:

rvclNone [1105]

See also:

- HoverBackColor [811];
- Color [810];
- HoverColor [812].

See also properties of TRVStyle:

- TextBackgroundKind [762].

Specifies default direction for the text of this style.

```
property BiDiMode: TRVBiDiMode [1058];
```

(introduced in version 1.6)

Value	Meaning
rvbdUnspecified	Text direction is inherited from higher level, i.e. from paragraph (see BiDiMode [831] 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 [137].

Horizontal character scaling value for text of this style, %

```
property CharScale: Integer;
```

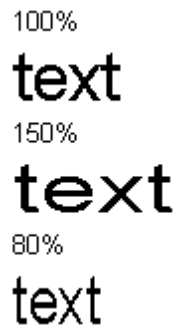
(Introduced in version 1.5)

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 [781]. Any changes of RichView's Canvas.TFont will reset font scaling to default.

Note: this property is ignored in FireMonkey version

Default value:
100

Specifies character set for text of this style.

```
property Charset: TRVFontCharset[1075];
```

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").

Note: this property exists in FireMonkey version, but is not used for drawing text.

Default value:
DEFAULT_CHARSET

Specifies intercharacter spacing for text of this style.

```
property CharSpacing: TRVStyleLength[1094];
```

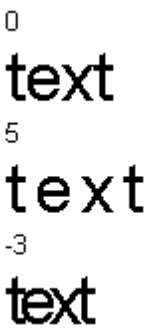
(introduced in version 1.6)

This property defines expansion (if positive) or compression (if negative) of space between characters.

This value is measured in Units [764] of TRVStyle [739] component which owns this text style.

Example:

This example shows text formatted with CharSpacing = 0, 5, and -3 pixels.



CharSpacing Example

Note: if TextEngine[763] is Windows API, all expanded characters are aligned to the left side (space is added to the right side of the characters).

if TextEngine[763] 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.

Note: this property is ignored in FireMonkey version

Default value:

0

Specifies color for text of this style.

```
property Color: TRVColor;
```

Default value:

rvclWindowText[1105]

See also:

- BackColor[808];
- HoverColor[812];
- HoverBackColor[811].

Specifies the width of a text item if its text is empty.

```
property EmptyWidth: TRVStyleLength[1094];
```

(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[808] for them.

For all normal text items, selection[112]/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[815].

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: TRVColor[1063];
```

(introduced in version 1.2)

If this style is a hypertext style (Jump[823]=*True*), background of text of this style can be highlighted with this color when user moves the mouse pointer above it. Set HoverBackColor=BackColor[808] to disable effect.

By default, RichViewEdit[469] also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[483] property.

This property has no effect for non-hypertext styles.

Default value:

rvclNone[1105]

See also properties:

- BackColor[808];
- HoverColor[812];
- Jump[823];
- JumpCursor[813];
- HoverEffects[812];
- HoverUnderlineColor[813].

See also properties of TRVStyle:

- HoverColor[750];
- FullRedraw[750];
- TextBackgroundKind[762].

Specifies highlight color for hypertext links of this style.

```
property HoverColor: TRVColor[1063];
```

If this style is a hypertext style (Jump[823]=*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 *rvclNone*[1105], TRVStyle.HoverColor[750] is used. If it is, in its order, equal to *rvclNone*[1105], no hover color effect is used.

By default, RichViewEdit[469] also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[483] property.

This property works only for hypertext styles, and for styles having *rvteHighlightable* in Options[814].

Default value:

rvclNone[1105]

See also properties:

- Color[810];
- HoverBackColor[811];
- Jump[823];
- JumpCursor[813];
- HoverEffects[812];
- HoverUnderlineColor[813].

See also properties of TRVStyle:

- HoverColor[750];
- FullRedraw[750].

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[823]=*True*), these effects are applied to text of this style when user moves the mouse pointer above it.

By default, RichViewEdit[469] also applies these effects to hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[483] property.

This property works only for hypertext styles, and for styles having *rvteHighlightable* in Options[814].

Effect	Meaning
rvheUnderline	Text is underlined.

This underline may have a custom color, see HoverUnderlineColor[813] property.

Default value:

[]

See also properties:

- Jump[823];
- HoverColor[812];
- HoverBackColor[811];
- JumpCursor[813].

See also properties of TRVStyle:

- FullRedraw[750].

Specifies underline highlight color for hypertext links of this style.

```
property UnderlineColor: TRVColor[1063];
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[817], or *rvheUnderline* is included HoverEffects[812].

If this style is a hypertext style (Jump[823]=*True*), underline of text of this style can be highlighted with this color when user moves the mouse pointer above it. This property works only for hypertext styles, and for styles having *rvteHighlightable* in Options[814].

If this property is equal to *rvclNone*[1105], UnderlineColor[818] is used. If it is, in its order, equal to *rvclNone*[1105], HoverColor[812] is used. If it is, in its order, equal to *rvclNone*[1105], Color[810] is used.

RTF and DocX Note:

This attribute cannot be stored in RTF and DocX files.

HTML Note:

Colored underlines are not supported in HTML without CSS, so TRichView.HTMLSaveProperties[244].HTMLSavingType[441] must be *rvhtmlstNormal*, otherwise this attribute will be ignored. TRichView saves colored underline as a bottom border side of the text's element.

Default value:

rvclNone[1105]

See also:

- UnderlineType[819].

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[823]=*True*).

Default value:

crJump

See also properties of TRVStyle:

- JumpCursor[752].

See also:

- Cursors used by components of TRichView family[147].

Options for exporting text of this style to RTF and HTML.

```
type
  TRVTextOption = (rvteoHTMLCode, rvteoRTFCode,
    rvteoDocXCode, rvteoDocXInRunCode,
    rvteoHidden, rvteHighlightable);
  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. A 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 [247]
rvteHighlightable	If set, text is highlighted below the mouse pointer, even if it is not a hyperlink, see HoverColor [812], HoverBackColor [811], HoverEffects [812] properties.

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 [350];
- TCustomRichView.SaveDocX [341];

- About HTML export [132].

Links this text style to the style template of paragraph.

```
property ParaStyleTemplateId: TRVStyleTemplateId[1094];
```

(introduced in version 14)

This property is used only if style templates are used [263].

This property is redundant: the style template of paragraph is linked using TParaInfo [837].StyleTemplateId [805]. However, it is necessary to distinguish text styles having all identical properties but used for text in paragraphs linked to different style templates.

TRichView ensures 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 [764], 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 [469]. Text of this style has special behavior when editing.

General Protection Options

Option	Meaning
rvprStyleProtect	Protects text from changing style with ApplyTextStyle [502], ApplyTextStyleTemplate [502], ApplyStyleTemplate [501]. This protection is ignored by ApplyStyleConversion [501], but you can take it into account manually in OnStyleConversion [595] event.
rvprStyleSplitProtect	Protects from applying text style to the part of text item. ApplyTextStyle [502] and ApplyStyleConversion [501] 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.

Option	Meaning
	Deletion depends on *rvoFastDeleteProtectedText* option in TRichViewEdit.EditorOptions [483].
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 [810] > 0, all characters can be deleted, but the item is not deleted. See also: *rvepDeleteProtect* item property [1067].
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 [482] 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 [583] 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 [833] of paragraph style;
- EmptyWidth [810] property

Demo projects:

- Demos*\Assorted\Fields\

The properties specify the font height.

```
property Size: TRVFontSize[1076];
property SizeDouble: TRVFontSize[1076];
```

(SizeDouble is introduced in version 15)

Size specifies the font size in points [144].

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.

In VCL and LCL version, **SizeDouble** allows to specify a font size with higher precision. In FireMonkey version, these values are floating point values, **SizeDouble** is maintained for compatibility with VCL version.

In VCL and LCL version, **Size** is similar to TFont.Size. In FireMonkey version, TFont.Size is measured in pixels, while **Size** is measured in points.

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 [818];

- HoverEffects[812].

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:

- capitalization works only under WinNT4 and newer.

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[817].

Default value:

rvsssNormal

See also properties:

- VShift[820].

Specifies underline color.

```
property UnderlineColor: TRVColor[1063];
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[817] (or *rvheUnderline* is included HoverEffects[812], under the mouse pointer).

If UnderlineColor=*rvclNone*[1105], text color is used for underline (Color[810] in normal state, HoverColor[812] for hyperlinks under the mouse pointer).

Values other than *rvclNone*[1105] define underline color. HoverUnderlineColor[813] 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 without CSS, so TRichView.HTMLSaveProperties[244].HTMLSavingType[441] must be *rvhtmlstNormal*, otherwise this attribute will be ignored. TRichView saves colored underline as a bottom border side of the text's element.

Default value:

rvclNone[1105]

See also:

- UnderlineType[819].

Specifies the style of underline.

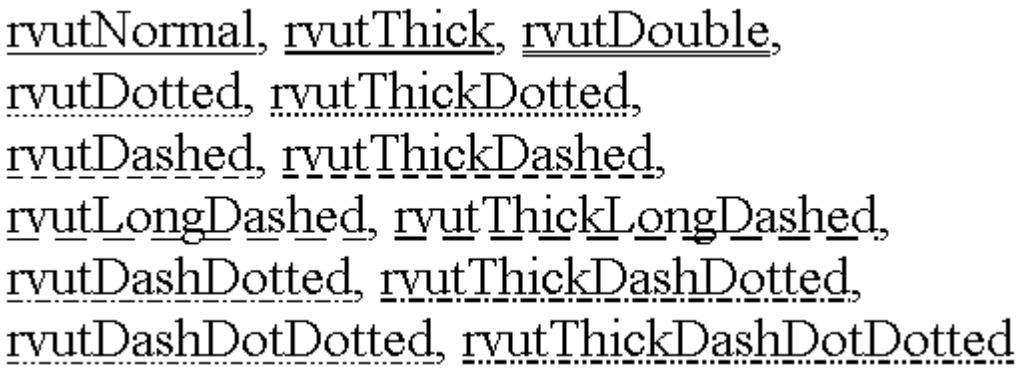
```
type
  TRVUnderlineType = (rvutNormal, rvutThick, rvutDouble,
    rvutDotted, rvutThickDotted,
    rvutDashed, rvutThickDashed,
    rvutLongDashed, rvutThickLongDashed,
    rvutDashDotted, rvutThickDashDotted,
    rvutDashDotDotted, rvutThickDashDotDotted);
property UnderlineType: TRVUnderlineType;
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[817] (or *rvheUnderline* is included HoverEffects[812], under the mouse pointer).

By default, underline color is defined in Color[810] property, but it can be altered by UnderlineColor[818] property (see also HoverColor[812]/HoverUnderlineColor[813] 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 without CSS, so TRichView.HTMLSaveProperties[244].HTMLSavingType[441] must be *rvhtmlstNormal*, otherwise this attribute will be ignored. TRichView saves custom underline as a bottom border side of the text's element.

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[818] (the effect is cumulative).

Default value:

0

See also:

- TRVExtraItemProperty[1067] type (*rvepVShift*)

6.2.7.5.2.2 Methods

In TCustomRVFontInfo

Assign[820]
AssignTo[820]
Create[821]
IsEqual[821]

Copies the contents of **Source** to this text style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TFont, TCustomRVInfo[803], or TCustomRVFontInfo to this text style.

See also:

- TFontInfo.Assign[825] (TFontInfo[821] 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[764][0]);
```

A constructor, creates and initializes a new instance of TCustomRVFontInfo.

```
constructor Create(Collection: TCollection);override;
```

The constructor initializes:

```
// Properties inherited from TCustomRVInfo[803]:
Standard[804]     := True;
StyleName[805]    := 'Font Style';
// Properties defined in TCustomRVFontInfo:
Size[817]         := 10;
FontName[811]     := 'Arial';
Color[810]        := clWindowText;
UnderlineColor[818] := clNone;
BackColor[808]    := clNone;
HoverColor[812]   := clNone;
HoverUnderlineColor[813] := clNone;
Charset[809]      := DEFAULT_CHARSET;
JumpCursor[813]   := crDefault;
CharScale[808]    := 100;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a text style in code, you should assign Standard[804] := *False*.

Compares this text style with **Value**.

```
function IsEqual(Value: TCustomRVFontInfo; IgnoreList: TRVFontInfoProperties[1075])
```

(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[1075].

The following properties are always ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

See also:

- TFontInfo.IsEqual[827] (TFontInfo[821] is inherited from this class);
- TCustomRVParaInfo.IsEqual[837].

6.2.7.5.3 TFontInfo

This class defines attributes of text items[167] in TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[616] controls.

This is an item in the collection of TRVStyle[739].TextStyles[764]. The class of the collection itself is TFontInfos[785].

Unit RVStyle.

Syntax

```
TFontInfo = class(TCustomRVFontInfo[806])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[803]
TCustomRVFontOrParaInfo[803]
TCustomRVFontInfo[806]

Properties

The most of properties are inherited from TCustomRVFontInfo[806].

The new properties are:

- Jump[823] – "is this a hyperlink?"
- NextStyleNo[824] – index of text style to apply after pressing Enter in editor (if style templates are not used[263]);
- ModifiedProperties[824] may (temporarily) contain a list of properties having values different from values defined in a style templates (of this text style[805] and paragraph[815]).

Methods

IsEqual[827] compares this text style with another one.

You can use Assign[825] method to assign:

- TFontInfo to TFontInfo;
- TCustomRVFontInfo[806] to TFontInfo and vice versa;
- TFont to TFontInfo and vice versa.

See Also

- TFontInfos[785] – class of collection of text styles;
- TParaInfo[828] – class of paragraph style;
- TRVListInfo[841] – class of list style.

6.2.7.5.3.1 Properties

In TFontInfo

- Jump[823]
- ModifiedProperties[824]
- NextStyleNo[824]

Derived from TCustomRVFontInfo[806]

- BackColor[808]

- BiDiMode 808
- CharScale 808
- Charset 809
- CharSpacing 809
- Color 810
- EmptyWidth 810
- FontName 811
- HoverBackColor 811
- HoverColor 812
- HoverEffects 812
- HoverUnderlineColor 813
- JumpCursor 813
- Options 814
- ParaStyleTemplateId 815
- Protection 815
- Size 817
- SizeDouble 817
- Style 817
- StyleEx 818
- SubSuperScriptType 818
- UnderlineColor 818
- UnderlineType 819
- VShift 820

Derived from TCustomRVFontOrParaInfo 803

- StyleTemplateId 805

Derived from TCustomRVInfo 803

- Standard 804
- StyleName 805

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 812;
- HoverBackColor 811;
- HoverUnderlineColor 813;
- HoverEffects 812;
- JumpCursor 813.

See also:

- RichView hypertext overview 97.

Demo projects:

- Demos*\Assorted\Hypertext\

This property may contain a list of properties not defined in style templates [762] of this style.

```
property ModifiedProperties: TRVFontInfoProperties[1075];
```

(introduced in version 14)

This property is used only if style templates are used [263].

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 [805];
- ParaStyleTemplateId [815].

See also properties of TParaInfo:

- ModifiedProperties [839].

See also methods of TRVStyleTemplate:

- UpdateModifiedTextStyleProperties [850];
- ApplyToTextStyle [849] (contains an example).

Defines the text style for following paragraphs.

```
property NextStyleNo: Integer;
```

This property may contain either an index in Style [260].TextStyles [764], or the value -1.

This property is used in RichViewEdit [469], only if UseStyleTemplates [263]=*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 [263]=*True*, this property is ignored. See the the description NextId [846] property of the paragraph's style template [815] about defining text properties for new paragraphs.

Default value:

-1

See also

- TParaInfo.NextParaNo [840].

```
property Unicode: Boolean;
```

Obsolete property. Always *True*.

Default value:

True

See also:

- Unicode in RichView [135].

6.2.7.5.3.2 Methods

In TFontInfo

Assign 825
Create 825
Draw 825
IsEqual 827

Derived from TCustomRVFontInfo 806

AssignTo 820

Copies the contents of **Source** to this text style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TFont, TCustomRVInfo 803, TCustomRVFontInfo 806, or TFontInfo to this text style.

Example:

```
// Copying TFont to text style:
MyRVStyle.TextStyles 764[0].Assign(MyEdit.Font);

// Copying text style to text style:
MyRVStyle.TextStyles 764[0].Assign(MyRVStyle.TextStyles 764[1]);
// or, the same:
MyRVStyle.TextStyles 764[0] := MyRVStyle.TextStyles 764[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 821, this methods initializes:

```
NextStyleNo 824     := -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 804 := *False*.

Draws **s** on **Canvas**.

```
procedure Draw(const s: TRVUnicodeString 1099; Canvas: TCanvas;
  ThisStyleNo: Integer; SpaceAtLeft, Left, Top, Width, Height
  {$IFDEF RVUSEBASELINE}
   ,BaseLine
  {$ELSE}
    {$IFDEF RVHIGHFONTS}
     ,DrawOffsY
    {$ENDIF}
```

```
  {$ENDIF}: TRVCoord[1064];
  TextWidth: TRVCoord[1064]; SpaceCount: Integer;
  RVStyle: TRVStyle[739];
  DrawState: TRVTextDrawStates[1097];
  Printing, PreviewCorrection, ForMetafile: Boolean;
  ColorMode: TRVColorMode[1063]; DarkMode: Boolean;
  DefBiDiMode: TRVBiDiMode[1058];
  RefCanvas: TCanvas; GraphicInterface: TRVGraphicInterface;
  LeftNoExpIndex, RightNoExpIndex: Integer;
  PSaD: PRVScreenAndDevice[1090]; AControl: TControl = nil);
```

(changed in versions 18, 19, 23)

Normally, this procedure is called from TRVStyle[739].OnDrawStyleText[781] 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[764].

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[739] 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[635], and TRVPrint.PreviewCorrection[882]=*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**).

DarkMode: if *True*, luminance (lightness) of colors will be inverted.

DefBiDiMode is a bi-di mode of paragraph containing this text.

RefCanvas – canvas used to measure text:

- when drawing in TRichView[216], TRichViewEdit[469], TRVReportHelper[721], TRVPrint[696], 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.

PSaD – pointer to a record used for conversion between TRVStyleLength[1094] values and device pixels. It may be *nil* when applying to the screen. This parameter is necessary to apply CharSpacing[809] correctly.

AControl – the editor where this text is drawn. This parameter is optional. It is necessary only in Delphi 10.4+, if **ColorMode** = *rvcmVCLThemeColor*, and the editor has StyleName property assigned. This parameter allows choosing a VCL theme specific for this editor instead of the global VCL theme.

Compares this text style with **Value**.

```
function IsEqual(Value: TCustomRVFontInfo; IgnoreList: TRVFontInfoProperties[1075])
```

(introduced in version 1.3)

Parameter:

Value – object to compare with this text style. The class of this object is TCustomRVFontInfo[806] or inherited from it. If **Value** is TFontInfo, values of NextStyleNo[824] 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[804];
- ModifiedProperties[824];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

See also:

- TCustomRVFontInfo.IsEqual[821] (method of the ancestor class);
- TFontInfos.FindSuchStyle[791];
- TParaInfo.IsEqual[841].

6.2.7.5.4 TCustomRVParaInfo

This is an ancestor class for TParaInfo[837] (class of item in collection of paragraph styles[757]).

TCustomRVParaInfo introduces the most of properties of paragraph style.

Unit RVStyle.

Syntax

```
TCustomRVParaInfo = class(TCustomRVFontOrParaInfo[803])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVFontOrParaInfo[803]
TCustomRVInfo[803]

Properties

- Alignment, LastLineAlignment[829] – alignment of paragraph (left, right, center, justify or distribute);
- FirstIndent[831] – indent of the first line of paragraph, summarized with LeftIndent;
- LeftIndent[832] – indent from left margin;
- RightIndent[835] – indent from right margin;
- SpaceBefore[836] – spacing above the paragraph;
- SpaceAfter[835] – spacing below the paragraph;
- Border[831] – border around the paragraph;
- Background[830] – background of paragraph;
- Tabs[836] – collection of tab stops;
- LineSpacing[832], LineSpacingType[833] – line spacing;
- BiDiMode[831] – default text direction;
- Options[833] – protection, wrapping.
- OutlineLevel[835] allows defining headings.

Methods

IsEqual[837] compares this paragraph style with another one.

You can use Assign[836] method to assign TCustomRVParaInfo to TCustomRVParaInfo.

See Also

Classes:

- TParaInfo[837] – class inherited from this class, a paragraph style;
- TParaInfos[791] – collection of text styles (TParaInfo[837]).

Overview:

- TRichView Paragraphs[100]

6.2.7.5.4.1 Properties

In TCustomRVParaInfo

- Alignment 829
- Background 830
- BiDiMode 831
- Border 831
- FirstIndent 831
- LastLineAlignment 829
- LineSpacing 832
- LineSpacingType 833
- LeftIndent 832
- Options 833
- OutlineLevel 835
- RightIndent 835
- SpaceAfter 835
- SpaceBefore 836
- Tabs 836

Derived from TCustomRVFontOrParaInfo 803

- StyleTemplateId 805

Derived from TCustomRVInfo 803

- Standard 804
- StyleName 805

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

Value	Meaning
	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.

* this value is not supported in FireMonkey version (in FireMonkey, it works like *rvaLeft*).

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[831], 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 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[851];
```

(Introduced in version 1.2)

See TRVBackgroundRect[851] for additional information.

Note: RTF and HTML support only equal Background.BorderOffsets[851] and Border[831].BorderOffsets[853]. RTF and DocX does not support Opacity[852].

See also:

- Paragraph Background[102].

Specifies default text direction for paragraphs of this style.

```
property BiDiMode: TRVBiDiMode[1058];
```

(introduced in version 1.6)

Value	Meaning
rvbdUnspecified	Text direction is inherited from higher level, i.e. from TCustomRichView (see BiDiMode[230])
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[137].

Defines a border around paragraphs of this style (none by default)

```
property Border:TRVBorder[852];
```

(Introduced in version 1.2)

See TRVBorder[852] for additional information.

Note: RTF and HTML support only equal Background[830].BorderOffsets[851] and Border.BorderOffsets[853].

See also:

- Paragraph Background[102].

Indent of the first line for paragraphs of this style.

```
property FirstIndent: TRVStyleLength[1094];
```

This value is measured in Units[764] of TRVStyle[739] 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[245]+LeftIndent[832]+FirstIndent.

In paragraphs with bullets or numbering, this property is overridden by FirstIndent[860] of list level.

Default value:

0

See also properties:

- LeftIndent[832], RightIndent[835];
- SpaceBefore[836], SpaceAfter[835].

See also properties of TRVListLevel:

- FirstIndent[860].

Left indent for paragraphs of this style.

```
property LeftIndent: TRVStyleLength[1094];
```

This value is measured in Units[764] of TRVStyle[739] component which owns this paragraph style.

Indent for paragraph from left border (assuming that indents are converted to pixels) = TCustomRichView.LeftMargin[245]+LeftIndent [+ FirstIndent[831] for the first line]

In paragraphs with bullets or numbering, this property is overridden with LeftIndent[864] of list level (if BiDiMode[831]<>*rvbdRightToLeft*).

Default value:

0

See also properties:

- RightIndent[835];
- SpaceBefore[836], SpaceAfter[835].

See also properties of TRVListLevel:

- LeftIndent[864].

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[833] 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 in Units[764] of TRVStyle[739] 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 [832] defines spacing in % of text height. Extra spacing is added/subtracted below lines.
rvlsSpaceBetween	LineSpacing [832] is a value which will be added between lines, measured in RVStyle.Units [764]. This type of line spacing cannot be exported in RTF and HTML. Extra spacing is added below lines.
rvlsLineHeightAtLeast	LineSpacing [832] defines minimal line height, measured in RVStyle.Units [764]. Extra spacing is added above and below lines.
rvlsLineHeightExact	LineSpacing [832] defines line height, measured in RVStyle.Units [764]. 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 [247]

Editing and Protection Options:

Option	Meaning
rvpaoReadOnly	Disallows editing paragraphs of this style in TRichViewEdit [469]. 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 [498], ApplyParaStyleTemplate [500], ApplyStyleTemplate [501] cannot change style of paragraphs of this style. This option does not protect from ApplyParaStyleConversion [499], but you can take it into account in OnParaStyleConversion [592] 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[815];
- TCustomRichView.WordWrap[265].
- TRVTableRow.KeepTogether[967]

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[1094];
```

This value is measured in Units[764] of TRVStyle[739] component which owns this paragraph style.

Indent for paragraph from the right border (assuming that indents are converted to pixels) = TCustomRichView.RightMargin[251]+RightIndent.

In paragraphs with bullets or numbering, this property is overridden with LeftIndent[864] of list level, if BiDiMode[831]=*rvbdRightToLeft*.

Default value:

0

See also properties:

- LeftIndent[832];
- SpaceBefore[836], SpaceAfter[835].

See also properties of TRVListLevel:

- LeftIndent[864].

Specifies the size of the spacing below the paragraph.

```
property SpaceAfter: TRVStyleLength[1094];
```

This value is measured in Units[764] of TRVStyle[739] 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[836];
- LeftIndent[832], RightIndent[835].

Specifies the size of the spacing above the paragraph.

```
property SpaceBefore: TRVStyleLength[1094];
```

This value is measured in Units[764] of TRVStyle[739] 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[835];
- LeftIndent[832], RightIndent[835].

Defines tab stops and their properties for paragraphs of this style.

```
property Tabs: TRVTabInfos[854];
```

(introduced in v1.9)

This is a sorted collection of TRVTabInfo[856] (sorted by Position[857] in ascending order).

Default value:

Empty collection.

See also properties of TRVStyle:

- DefTabWidth[745].

6.2.7.5.4.2 Methods

In TCustomRVParaInfo

Assign[836]
Create[836]
IsEqual[837]

Copies the contents of **Source** to this paragraph style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TCustomRVInfo[803] or TCustomRVParaInfo to this paragraph style.

See also:

- TParaInfo.Assign[840] (TParaInfo[837] 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[803]:
Standard[804]         := True;
StyleName[805]        := 'Paragraph Style';
// Properties defined in TCustomRVParaInfo:
Alignment[829]        := rvaLeft;
LineSpacingType[833]  := 100;
```

```
LineSpacing[832]      := rvlsPercent;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a paragraph style in code, you should assign Standard[804] := *False*.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties[1083]): 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[1083].

The following properties are always ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default)).

See also:

- TParaInfo.IsEqual[841] (TParaInfo[837] is inherited from this class);
- TCustomRVFontInfo.IsEqual[821].

6.2.7.5.5 TParaInfo

This is a paragraph style. It defines attributes of paragraphs[100] in TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[616] controls.

This is an item in the collection of TRVStyle[739].ParaStyles[757]. The class of the collection itself is TParaInfos[791].

Unit RVStyle.

Syntax

```
TParaInfo = class(TCustomRVParaInfo[828])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[803]
TCustomRVFontOrParaInfo[803]
TCustomRVParaInfo[828]

Properties

The most of properties are inherited from TCustomRVParaInfo[828].

The new properties are:

- DefStyleNo [839] – default text style for paragraph;
- NextParaNo [840] – index of paragraph style to apply after pressing **Enter** in editor (if style templates are not used [263]);
- ModifiedProperties [839] may (temporarily) contain a list of properties having values different from values defined in a style template [805].

Methods

IsEqual [841] compares this paragraph style with another one.

You can use Assign [840] method to assign:

- TParaInfo to TParaInfo;
- TCustomRVParaInfo [828] to TParaInfo and vice versa.

See Also

- TParaInfos [791] – class of collection of paragraph styles;
- TFontInfo [821] – class of text style;
- TRVListInfo [841] – class of list style;
- RichView Paragraphs [100].

6.2.7.5.5.1 Properties

In TParaInfo

- DefStyleNo [839]
  ModifiedProperties [839]
- NextParaNo [840]

Derived from TCustomRVParaInfo [828]

- Alignment [829]
- Background [830]
- BiDiMode [831]
- Border [831]
- FirstIndent [831]
- LastLineAlignment [829]
- LineSpacing [832]
- LineSpacingType [833]
- LeftIndent [832]
- Options [833]
- RightIndent [835]
- SpaceAfter [835]
- SpaceBefore [836]
- Tabs [836]

Derived from TCustomRVFontOrParaInfo [803]

- StyleTemplateId [805]

Derived from TCustomRVInfo [803]

- Standard [804]
- StyleName [805]

Specifies the default text style for paragraphs of this style.

```
property DefStyleNo: Integer;
```

(introduced in version 1.8)

Deprecated! Use style templates [762] to link text and paragraph properties together.

You can use the rvsDefStyle [1127] constant (= MaxInt, a large positive integer value) as a StyleNo parameter of AddNL [277] 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 [310] method and CurTextStyleNo [482] return a translated text style (they never return rvsDefStyle).

Use GetItem [304](i).StyleNo and ActualCurTextStyleNo [480] to get the text style as is.

Default value:

-1

This property may contain a list of properties not defined in the style template [762] of this style.

```
property ModifiedProperties: TRVParaInfoProperties [1083];
```

(introduced in version 14)

This property is used only if style templates are used [263].

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 [805].

See also properties of TFontInfo:

- ModifiedProperties [824].

See also methods of TRVStyleTemplate:

- UpdateModifiedParaStyleProperties [850];
- ApplyToParaStyle [848] (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[260].ParaStyles[757], or the value -1.

This property is used in RichViewEdit[469], only if UseStyleTemplates[263]=*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[263]=*True*, this property is ignored, NextId[846] property of the paragraph's style template[805] is used instead.

Default value:

-1

See also:

- TFontInfo.NextStyleNo[824].

6.2.7.5.5.2 Methods

In TParaInfo

Assign[840]
Create[840]
IsEqual[841]

Copies the contents of **Source** to this paragraph style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TCustomRVInfo[803], TCustomRVParaInfo[828], or TParaInfo to this paragraph style.

Example:

```
// Copying paragraph style to paragraph style:
MyRVStyle.ParaStyles[757][0].Assign(MyRVStyle.ParaStyles[757][1]);
// or, the same:
MyRVStyle.ParaStyles[757][0] := MyRVStyle.ParaStyles[757][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[836], this methods initializes:

```
DefStyleNo[839] := -1;
NextParaNo[840] := -1;
```

Note: usually, when you create a paragraph style in code, you should assign Standard[804] := *False*.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties[1083]): 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[1083].

The following properties are always ignored when comparing styles:

- Standard[804];
- StyleName[805] (if RichViewCompareStyleNames[1111]=*False* (default));
- ModifiedProperties[839].

See also:

- TCustomRVParaInfo.IsEqual[837] (method of the ancestor class);
- TParaInfos.FindSuchStyle[794];
- TFontInfo.IsEqual[827].

6.2.7.5.6 TRVListInfo

This is a paragraph list style. It defines attributes of list markers[180] in TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[616] controls.

This is an item in the collection of TRVStyle[739].ListStyles[755]. The class of the collection itself is TRVListInfos[795].

Unit RVStyle.

Syntax

```
TRVListInfo = class(TCustomRVInfo[803])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[803]

Properties

- Levels[842] – collection of list levels (TRVListLevel[859])
- OneLevelPreview[842] suggests appearance of thumbnail.

Other properties are inherited from TCustomRVInfo[803].

Methods

- AllNumbered[842] returns *True* if all levels represent numbered lists;

- HasNumbering [843] returns *True* if at least one of list levels represents a numbered list.

See Also

- TRVListInfos [795] – class of collection of paragraph list styles;
- TFontInfo [821] – class of text style;
- TParaInfo [828] – class of paragraph style;
- List markers item type [180].

6.2.7.5.6.1 Properties

In TRVListInfo

- Levels [842]
- OneLevelPreview [842]

Derived from TCustomRVInfo [803]

- Standard [804]
- StyleName [805]

Levels of paragraph list style, collection of TRVListLevel [859] objects.

```
property Levels: TRVListLevelCollection[858];
```

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 [842].

6.2.7.5.6.2 Methods

In TRVListInfo

AllNumbered [842]
HasNumbering [843]

Returns *True* if all levels [842] represent numbered lists

```
function AllNumbered: Boolean;
```

See also:

- HasNumbering [843].

Returns *True* if at least one of list levels [842] represents a numbered list.

```
function HasNumbering: Boolean;
```

See also:

- AllNumbered [842].

6.2.7.5.7 TRVStyleTemplate

This class represents a "real" style controlling properties of TFontInfo [821] and TParaInfo [821] objects.

This is an item in the collection of TRVStyle [739].StyleTemplates [762]. The class of the collection itself is TRVStyleTemplateCollection [797].

Unit RVStyle.

Syntax

```
TRVStyleTemplate = class(TCollectionItem)
```

(introduced in version 14)

Hierarchy

TObject
TPersistent
TCollectionItem

Main Properties

- Kind [845] – kind of this style templates (defines it applicability to text or paragraphs);
- Name [845] – unique name;
- Id [844] – unique identifier;
- TextStyle [847], ValidTextProperties [848] – text properties;
- ParaStyle [846], ValidParaProperties [847] – paragraph properties;
- ParentId [847] – a reference to a parent style template;
- NextId [846] – a reference to a style template for following paragraphs;
- QuickAccess [847] – recommends to provide a quick access to this style template in applications UI.

Main Methods

- UpdateModifiedTextStyleProperties [850] fills the set of properties of the specified text style that were changed after applying this style template;
- UpdateModifiedParaStyleProperties [850] fills the set of properties of the specified paragraph style that were changed after applying this style template;
- ApplyToTextStyle [849] applies this style template to the specified text style;
- ApplyToParaStyle [848] applies this style template to the specified paragraph style.

Linking

Each style template has two properties having unique values: Id [844] and Name [845]. 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 [805] from TextStyles [764], ParaStyles [757], and between StyleTemplates are made via Id property.

Name is used when merging style templates of two different documents.

Unlike items in TRVStyle.TextStyles[764] and TRVStyle.ParaStyles[757], 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[501]).

See also

- Styles and style templates[103].

6.2.7.5.7.1 Properties

In TRVStyleTemplate

- Id[844]
- Kind[845]
- Name[845]
- Next[846]
- NextId[846]
- ParaStyle[846]
- Parent[847]
- ParentId[847]
- QuickAccess[847]
- TextStyle[847]
- ValidParaProperties[847]
- ValidTextProperties[848]

An identifier of this style template.

```
property Id: TRVStyleTemplateId[1094];
```

Value of this property is generated automatically using a random number generator. Each item has an identifier that is unique in the collection[797].

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[799] with CopyIds=*True*).

This property is used to refer to this style template within a single TRVStyle[739] component:

- another style template may refer to this style-template using ParentId[847] property;
- another style template may refer to this style-template using NextId[846] property;
- text[764] and paragraph[757] styles of the same TRVStyle may refer to this style template using StyleTemplateId[805] property.

Why do style templates need two unique properties, **Id** and Name[845]? 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[797]:

- FindById, FindItemById[800].

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 [757] (i.e. paragraph styles can be linked [805] to this style template)
rvstkParaText	This style template can be applied to text styles [764] (i.e. text styles can be linked [805] to this style template)

Assigning a new value to Name [845] 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 [845] 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[185]/endnote[184] characters, and to note references[190];
- "endnote text", "footnote text" to apply to text in Document[984] of new footnote/endnote.

See also members of TRVStyleTemplateCollection[797]:

- FindByName, FindItemByName[800];
- DefStyleName[798].

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[1094];
property Next: TRVStyleTemplate; // read-only
```

This property is used only for style templates applied to paragraphs, so it is ignored if Kind[845] =*rvstkText*.

To make some style template a next style template, assign its Id[844] to **NextId** of this style template.

NextId equals to Id[844] 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[844]=**NextId**.

When Enter is pressed, and style templates are used[263]:

- 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[823]=*True*), the new text is not linked to any style template, its attributes are changed accordingly, its Jump[823]=*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[764] and ParaStyles[757] 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[828]);
property ParaStyle: TRVSTParaInfo;
```

ParaStyle is ignored if Kind[845]=*rvstkText*

Only sub-properties listed in ValidParaProperties[847] 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[1094];
property Parent: TRVStyleTemplate; // read-only
```

To make some style template a parent of this style template, assign its Id[844] 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[844]=**ParentId**.

If Kind[845]=*rvstkText*, only style templates having Kind[845]=*rvstkText* can be its parents.

If Kind[845]<>*rvstkText*, only style templates having Kind[845]<>*rvstkText* can be its parents.

Circular references are not allowed.

The style template inherits (from parents) properties not listed in its ValidTextProperties[848] and ValidParaProperties[847].

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[806]);
property TextStyle: TRVSTFontInfo;
```

Only sub-properties listed in ValidTextProperties[848] are taken into account. Values of other sub-properties are ignored.

A set of properties of ParaStyle[846] defined in this style template.

```
property ValidParaProperties: TRVParaInfoProperties[1083];
```

Only properties included in this set are used. Values of other ParaStyle[846]'s properties are ignored.

This property is used only if Kind[845]<>*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 [847].

A set of properties of TextStyle [847] defined in this style template.

```
property ValidTextProperties: TRVFontInfoProperties[1075];
```

Only properties included in this set are used. Values of other TextStyle [847]'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 [847].

6.2.7.5.7.2 Methods

In TRVStyleTemplate

ApplyToParaStyle [848]
ApplyToTextStyle [849]
UpdateModifiedParaStyleProperties [850]
UpdateModifiedTextStyleProperties [850]

Changes properties of **AParaStyle** according to the properties of this style template.

```
procedure ApplyToParaStyle(AParaStyle: TCustomRVParaInfo[828]);
```

The method changes only properties of **AParaStyle** not listed in **AParaStyle**.ModifiedProperties [839]. If you want to change all properties, assign **AParaStyle**.ModifiedProperties [839]=[] before calling this method. If you want to change only properties controlled by style templates, call UpdateModifiedParaStyleProperties [850] before calling this method.

Applying a style template means applying properties of its ParaStyle [846] listed in ValidParaProperties [847], and properties inherited from Parent [847].

Kind [845] 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 [805] = Id [844]. This method may be called for *nil*-objects; in this case, it just assigns **AParaStyle**.StyleTemplateId [805] = -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[843];
begin
  RVStyle.ListStyles[755].Clear;
  RVStyle.TextStyles[764].Clear;
```

```
  RVStyle.TextStyles.Add;
  RVStyle.ParaStyles[757].Clear;
  RVStyle.ParaStyles.Add;
  StyleTemplate := RVStyle.StyleTemplates[762].NormalStyleTemplate[799];
  if StyleTemplate<>nil then begin
    RVStyle.ParaStyles[0].ModifiedProperties[839] := [];
    StyleTemplate.ApplyToParaStyle(RVStyle.ParaStyles[0]);
    RVStyle.TextStyles[0].ModifiedProperties[824] := [];
    TRVStyleTemplate(nil).ApplyToTextStyle[849](
      RVStyle.TextStyles[0], StyleTemplate);
    RVStyle.TextStyles[0].ParaStyleTemplateId[815] := StyleTemplate.Id[844];
  end;
end;
```

See also:

- ApplyToTextStyle[849]

Changes properties of **ATextStyle** according to the properties of this style template.

```
procedure ApplyToTextStyle(ATextStyle: TCustomRVFontInfo[806];
  ParaStyleTemplate: TRVStyleTemplate);
```

The method changes only properties of **ATextStyle** not listed in **ATextStyle**.ModifiedProperties[824]. If you want to change all properties, assign **ATextStyle**.ModifiedProperties[824]=[] before calling this method. If you want to change only properties controlled by style templates, call UpdateModifiedTextStyleProperties[850] before calling this method.

Applying a style template means applying properties of its TextStyle[847] listed in ValidTextProperties[848], and properties inherited from Parent[847].

Kind[845] 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[805] = Id[844]. This method may be called for *nil*-objects; in this case, it just assigns **ATextStyle**.StyleTemplateId[805] = -1.
2. Then it applies **ParaStyleTemplate** (if not *nil*) to remaining properties. The method does NOT assign **ATextStyle**.ParaStyleTemplateId[815].
3. Then it resets remaining properties of **ATextStyle** to default values.

Example: inserting a hyperlink in rve:TRichViewEdit[469]. If possible, this hyperlink is formatted using 'Hyperlink' style template.

```
var
  OldStyleTemplate, NewStyleTemplate, ParaStyleTemplate: TRVStyleTemplate[843];
  TextStyle: TFontInfo[821];
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.ParaStyleTemplateId);
    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', 'https://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[849] and ApplyToParaStyle[848] methods.

6.2.7.6 Classes of properties of a paragraph style

Classes of Properties of a Paragraph Style

The following classes are used as types of properties of TCustomRVParaInfo[828]:

- TRVBackgroundRect[851] – a type of Background[830] property, properties of paragraph background;
- TRVBorder[852] – a type of Border[831] property, properties of paragraph border;
- TRVTabInfos[854] – a type of Tabs[836] property, a collection of TRVTabInfo[856] items, defining properties of tab stops.

6.2.7.6.1 TRVBackgroundRect

This class defines background of paragraph (colored rectangle). This is a class of Background[830] property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBackgroundRect = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- BorderOffsets[851];
- Color[852];
- Opacity[852].

See also:

- TCustomRVParaInfo.Background[830];
- Paragraph Borders and Background[102].
- TRVBoxBackgroundRect[995] (background of a floating box)

6.2.7.6.1.1 Properties

In TRVBackgroundRect

- BorderOffsets[851]
- Color[852]
- Opacity[852]

Distance between the paragraph content and the border of the colored area.

```
property BorderOffsets: TRVRect[1038];
```

Sides of this rectangle are measured in Units[764] of TRVStyle[739] component which owns the paragraph style.

Lager positive values increase padding. Negative values force border to be inside the paragraph content.

Warning:

- Background[830].BorderOffsets.Top must not exceed SpaceBefore[836] property of the same paragraph.
- Background.BorderOffsets.Bottom must not exceed SpaceAfter[835] property of the same paragraph.
- Background.BorderOffsets.Left should not exceed LeftIndent[832] property of the same paragraph.
- Background.BorderOffsets.Right should not exceed RightIndent[835] property of the same paragraph.

Default value:

0,0,0,0 (border is drawn around the paragraph without gaps)

See also:

- Paragraph Background[102]

Color of paragraph background.

```
property Color: TRVColor[1063]
```

Default value:

rvclNone[1105] (transparent background)

Opacity of paragraph background

```
property Opacity: TRVOpacity[1082]
```

(introduced in version 16)

This property is used if Color[852]<>*clNone*.

Default value:

100000 (i.e. 100%)

6.2.7.6.2 TRVBorder

This class defines border around paragraph. This is a class of Border[831] property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBorder = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- BorderOffsets[853] – distance between paragraph content and border;
- Color[853] – border color;
- InternalWidth[853] – distance between border lines (for double and triple borders);
- Style[854] – border style;
- VisibleBorders[854] hides/shows border sides;
- Width[854] – line width.

See also:

- TCustomRVParaInfo.Border[831];
- Paragraph Borders and Background[102].

6.2.7.6.2.1 Properties

In TRVBorder

- BorderOffsets [853]
- Color [853]
- InternalWidth [853]
- Style [854]
- VisibleBorders [854]
- Width [854]

Spacing between paragraph content and border.

```
property BorderOffsets: TRVRect [1038];
```

Sides of this rectangle are measured in Units [764] of TRVStyle [739] component which owns the paragraph style.

Lager positive values increase spacing. Negative values force the border to be inside the paragraph content.

Warning:

- Border [831].BorderOffsets.Top must not exceed SpaceBefore [836] property of the same paragraph.
- Border.BorderOffsets.Bottom must not exceed SpaceAfter [835] property of the same paragraph.
- Border.BorderOffsets.Left should not exceed LeftIndent [832] property of the same paragraph.
- Border.BorderOffsets.Right should not exceed RightIndent [835] property of the same paragraph.

Default value:

0,0,0,0

See also:

- Paragraph Borders [102].

Color of border

```
property Color: TRVColor [1063]
```

This property cannot be used to hide the border. To hide border, set Style [854] =*rvbsNone*.

Default value:

rvclWindowText [1105]

Spacing between lines in double and triple borders.

```
property InternalWidth: TRVStyleLength [1094];
```

This value is measured in Units [764] of TRVStyle [739] component which owns this paragraph style.

For example, if Style [854] =*rvbDouble*, InternalWidth=1, and Width [854] =1, and RVStyle.Units [764] =*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 [764] =*rvstuPixels*, otherwise it could be too small.

Style of a border

```
property Style: TRVBorderStyle[1059];
```

See TRVBorderStyle[1059] for possible values.

Default value:

rvbNone (no border)

Defines which sides of border are visible.

```
property VisibleBorders: TRVBooleanRect[1018];
```

Default value:

True, *True*, *True*, *True* (all sides are visible, but border is still invisible because its Style[854]=*rvbNone* by default)

Width of border lines.

```
property Width: TRVStyleLength[1094];
```

This value is measured in Units[764] of TRVStyle[739] 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[854]=*rvbSingle*.

Thick lines (Style=*rvbThickInside* or *rvbThickOutside*) have double width.

Default value:

1

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

6.2.7.6.3 TRVTabInfos

Collection of tab stops (TRVTabInfo[856]) and their properties.

This is a type of TRVStyle[739].ParaStyles[757][i].Tabs[836] 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[857] 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[857], it is moved to the proper place.

Main properties:
- Items [855];
- Count.

6.2.7.6.3.1 Properties

In TRVTabInfos

Items [855]

Derived from TCollection

Count

Provides indexed access to the items in the collection of tab stops for the paragraph.

```
property Items[Index:Integer]: TRVTabInfo[856];
```

This is the default property of TRVTabInfos, so you can write

```
MyRVStyle.ParaStyles[757][i].Tabs[836][j]
```

instead of

```
MyRVStyle.ParaStyles[i].Tabs.Items[j].
```

6.2.7.6.3.2 Methods

In TRVTabInfos

AddFrom [855]
DeleteList [855]
Find [855]
Intersect [856]

You can also use methods derived from TCollection

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 [857] property)

The collection remains sorted.

Deletes tab stops at the specified **Positions**

```
procedure DeleteList(Positions: TRVIntegerList[1036]);
```

All tab stops having Position [857] 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 Position [857] 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: TRVTabInfos[854]);
```

6.2.7.6.4 TRVTabInfo

Defines properties for a tab stop. This is a type of item in TRVStyle[739].ParaStyles[757][i].Tabs[836] collection.

Unit RVStyle.

Syntax

```
TRVTabInfo = class(TCollectionItem)
```

(introduced in version 1.9)

Hierarchy

TObject
TPersistent
TCollectionItem

Properties

Position[857] – distance of this tab stop from LeftMargin[245];
Align[856] – alignment of tab relating to the followed text;
Leader[857] – text to fill this tabulator.

6.2.7.6.4.1 Properties

In TRVTabInfo

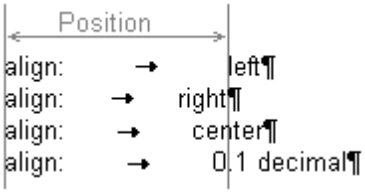
- Align[856]
- Leader[857]
- Position[857]

Defines a tab stop alignment (relative to the text after this tab stop)

```
type
  TRVTabAlign = (
    rvtaLeft, rvtaRight,
    rvtaCenter, rvtaDecimal);

property Align: TRVTabAlign;
```

Image:



Alignments are inverted for right-to-left paragraphs[137].

rvtaDecimal aligns content at the position of a decimal separator. It may be comma or semicolon character. Both these characters are treated as separators if they follow after a digit character. If not, they are treated as separators only if they are equal to the system default decimal separator (FormatSettings.DecimalSeparator).

Default value:

rvtaLeft

See also:

- Position [857].

Repeated text to fill the tabulator space.

```
property Leader: TRVUnicodeString[1099];
```

(changed in version 18)

Only the following values of this property can be saved in RTF and DocX:

Character	Description
.	dot
-	hyphen
_	underscore character
·	#$00B7, middle dot
=	equals sign

Default value:

'' (empty string)

Defines distance of this tab stop from LeftMargin [245].

```
property Position: TRVStyleLength[1094];
```

This value is measured in Units [764] of TRVStyle [739] component which owns the paragraph style with this tab stop.

For right-to-left paragraphs [137], the distance is measured from RightMargin [251].

Collections of tab stops (TRVStyle [739].ParaStyles [757][i].Tabs [836]) 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[836][i].Position := 10;
// index of this tab stop may be changed!
ParaStyle.Tabs[836][i].Align[856] := rvtaRight;
```

Correct:

```
ParaStyle.Tabs[836][i].Align[856] := rvtaRight;
ParaStyle.Tabs[836][i].Position := 10;
```

or

```
with ParaStyles.Tabs[836][i] do
begin
  Position := 10;
  Align[856] := rvtaRight;
end;
```

Default value:

0

6.2.7.7 Classes of properties of a list style

Classes of Properties of a List Style

The following classes are used as types of properties of TRVListInfo[841]:

- TRVListLevelCollection[858] – a type of Levels[842] property, a collection of TRVListLevel[859] items, defining properties of one level of a paragraph list (a bullet or a number)

6.2.7.7.1 TRVListLevelCollection

Collection of list levels. Each level (TRVListLevel[859]) defines properties for bullet or numbering of paragraphs.

This is a class of Levels[842] property of list styles[841].

Unit RVStyle.

Syntax

```
TRVListLevelCollection = class(TCollection)
```

Hierarchy

TObject
TPersistent
TCollection

Main Properties

Items[859] – array of TRVListLevel[859].

6.2.7.7.1.1 Properties

In TRVListLevelCollection

Items[859]

Derived from TCollection

Count

Provides indexed access to the items in the collection of paragraph list levels (bullets and numbering)

```
property Items[Index:Integer] : TRVListLevel[859];
```

Each level defines properties for bullet or numbering of paragraphs.

6.2.7.7.1.2 Methods

In TRVListLevelCollection

Add[859]

You can also use methods derived from TCollection

Adds a new style to the collection of list levels.

```
function Add: TRVListLevel[859];
```

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[859] array.

Add returns the new collection item.

6.2.7.7.2 TRVListLevel

One level of paragraph list, defines properties for bullet or numbering. This is a class of item in the Levels[842] collection of list style[841].

Unit RVStyle.

Syntax

```
TRVListLevel = class(TCollectionItem)
```

Hierarchy

TObject
TPersistent
TCollectionItem

Properties

Layout:

- FirstIndent[860], LeftIndent[864] override the corresponding properties of paragraph style;
- MarkerIndent[866] – indent of marker;
- MarkerAlignment[866] – position of marker relative to MarkerIndent.

List marker:

- ListType[864] – type of marker (bullet, number, image, etc.);
- Font[861] – font for text markers;

- Picture [868] – picture, used if ListType [864] =rvlstPicture;
- ImageList [863], ImageIndex [863] – used if ListType [864] in [rvlstImageList, rvlstImageListCounter];
- ImageWidth, ImageHeight [863] – used in FireMonkey if ListType [864] in [rvlstImageList, rvlstImageListCounter];
- FormatString [861] – format string for displaying text markers.

Numbering:

- StartFrom [868] – initial counter value for numbered lists;
- Options [867] – numbering options for this list level.

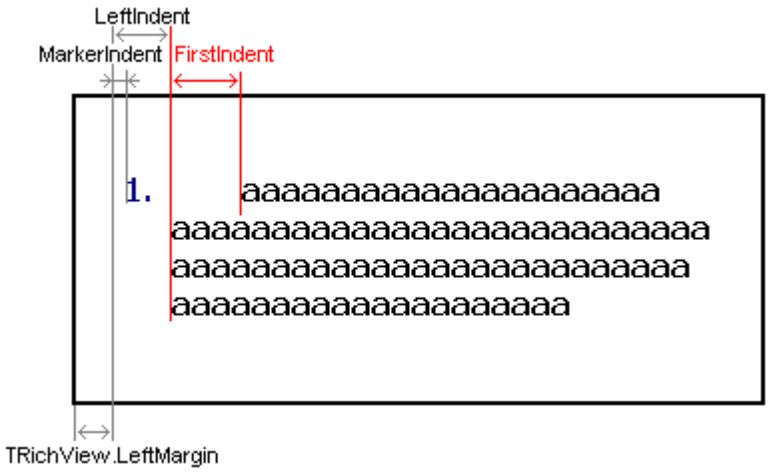
6.2.7.7.2.1 Properties

In TRVListLevel

- FirstIndent [860]
- Font [861]
- FormatString [861]
- ImageHeight [863]
- ImageIndex [863]
- ImageList [863]
- ImageWidth [863]
- LeftIndent [864]
- ListType [864]
- MarkerAlignment [866]
- MarkerIndent [866]
- Options [867]
- Picture [868]
- StartFrom [868]

Indent for the first line of paragraph text, relative to LeftIndent [864].

```
property FirstIndent: TRVStyleLength[1094];
```



FirstIndent Example

This value is measured in Units [764] of TRVStyle [739] component which owns the list style.

This value can be negative.

This property overrides FirstIndent[831] 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[764]=*rvstuPixels*, otherwise it could be too small.

See also properties:

- LeftIndent[864],
- MarkerIndent[866].

See also properties of TCustomRichView:

- LeftMargin[245].

Font for text markers.

```
type
 TRVMarkerFont = class (TRVFont[1032]);

 property Font: TRVMarkerFont;
```

This font is used for the following list types[864]:

- *rvlstBullet,*
- *rvlstDecimal,*
- *rvlstDecimalLeadingZero,*
- *rvlstLowerAlpha,*
- *rvlstUpperAlpha,*
- *rvlstLowerRoman,*
- *rvlstUpperRoman,*
- *rvlstLowerGreek,*
- *rvlstUnicodeBullet.*

Default value:

'Arial', 8

Defines format for text list markers.

```
property FormatString: TRVUnicodeString[1099];
```

This property is used for the following list types[864]:

- *rvlstBullet,*
- *rvlstDecimal,*
- *rvlstDecimalLeadingZero,*
- *rvlstLowerAlpha,*
- *rvlstUpperAlpha,*
- *rvlstLowerRoman,*
- *rvlstUpperRoman,*
- *rvlstLowerGreek.*

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[864]=*rvlstUnicodeBullet*, this text is displayed.

Since version 18, this property does not make sense, because FormatString[861] is Unicode as well.

Default value:

'' (empty string)

See also properties:

- FormatString[861].

Image index for bullets that show image from the image list

```
property ImageIndex: Integer;
```

If ListType[864]=*rvlstImageList*, the marker of this list level is the ImageIndex-th image from ImageList[863].

If ListType[864]=*rvlstImageListCounter*, the marker of this list level is the (Counter+ImageIndex-1)-th image from ImageList[863], there Counter is a value of counter for the given list marker.

FireMonkey: the component chooses the image that fits in ImageWidth x ImageHeight[863].

List counter is numbered from StartFrom[868].

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[868].

Image list for image of list marker

```
property ImageList: TCustomImageList;
```

ImageList is used If ListType[864] is *rvlstImageList* or *rvlstImageListCounter*.

When saving lists of this type in RVF[129], RichView stores value of ImageList.Tag. When loading, OnRVFImageListNeeded[400] occurs.

List level does not own this imagelist: it does not create and does not destroy it. It just has a link to it.

FireMonkey: the component chooses the image that fits in ImageWidth x ImageHeight[863].

See also properties:

- ImageIndex[863];
- ImageWidth, ImageHeight[863] [FMX];
- Picture[868].

Desired size of the list marker image.

```
property ImageWidth: TRVStyleLength[1094];
property ImageHeight: TRVStyleLength[1094];
```

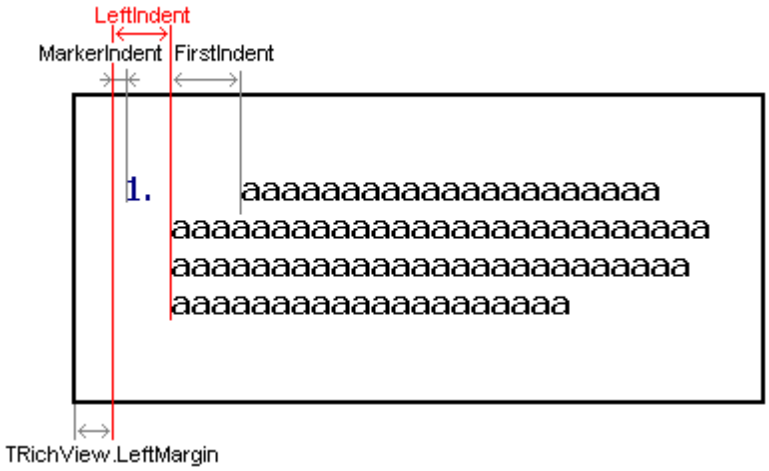
Currently, these properties are used only in FireMonkey, if ListType[864] = *rvlstImageList* or *rvlstImageListCounter*. The editor chooses the image that fits in the specified size. Zero values are interpreted as 16 device pixels.

See also properties:

- ImageIndex[863];
- ImageList[863].

Left indent of paragraph text.

```
property LeftIndent: TRVStyleLength[1094];
```



LeftIndent Example

This value is measured in Units[764] of TRVStyle[739] component which owns the list style.

This value can be negative, but it's not recommended.

This property overrides LeftIndent[832] of paragraph style for bulleted/numbered paragraphs. For right-to-left paragraphs[137], list marker is positioned to the right side of paragraph, and this property overrides RightIndent[835] of paragraph style instead.

Default value:

0

See also properties:

- FirstIndent[860],
- MarkerIndent[866].

See also properties of TCustomRichView:

- LeftMargin[245].

Type of list marker for this list level

```
type
  TRVListType = (rvlstBullet, rvlstPicture,
    rvlstImageList, rvlstDecimal,
    rvlstDecimalLeadingZero,
    rvlstLowerAlpha, rvlstUpperAlpha,
    rvlstLowerRoman, rvlstUpperRoman,
    rvlstLowerGreek,
    rvlstImageListCounter,rvlstUnicodeBullet);

property ListType: TRVListType;
```

Unnumbered List Types

List Type	Meaning
rvlstBullet	Unnumbered text (from FormatString[861] property). See also Font[861].
rvlstPicture	Picture (from Picture[868] property)
rvlstImageList	Image from an image list (ImageList[863] and ImageIndex[863] properties). **FireMonkey:** the component chooses the image that fits in ImageWidth x ImageHeight[863].

Numbered List Types

List Type	Meaning
rvlstDecimal	1, 2, 3,... (see FormatString[861], Font[861])
rvlstDecimalLeadingZero	01, 02, 03, ... (see FormatString[861], Font[861])
rvlstLowerAlpha	a, b, c, ..., z, aa, ab, ... (see FormatString[861], Font[861])
rvlstUpperAlpha	A, B, C, ..., Z, AA, AB, ... (see FormatString[861], Font[861])
rvlstLowerRoman	i, ii, iii, iv, ... (see FormatString[861], Font[861])
rvlstUpperRoman	I, II, III, IV, ... (see FormatString[861], Font[861])
rvlstLowerGreek	α, β, γ, ..., ω, αα, αβ, ... (see FormatString[861], Font[861])
rvlstImageListCounter	Image from an image list, depending on a counter (ImageList[863] and ImageIndex[863] properties). **FireMonkey:** the component chooses the image that fits in ImageWidth x ImageHeight[863].

Obsolete List Types

List Type	Meaning
rvlstUnicodeBullet	Unnumbered Unicode text (from FormatStringW[862] property). Obsolete value. Use *rvlstBullet*.

Default value:

rvlstBullet

Defines marker alignment relative to the MarkerIndent [866]

```
type
  TRVMarkerAlignment = (rvmaLeft, rvmaRight, rvmaCenter);

property MarkerAlignment: TRVMarkerAlignment;
```

MarkerAlignment Example

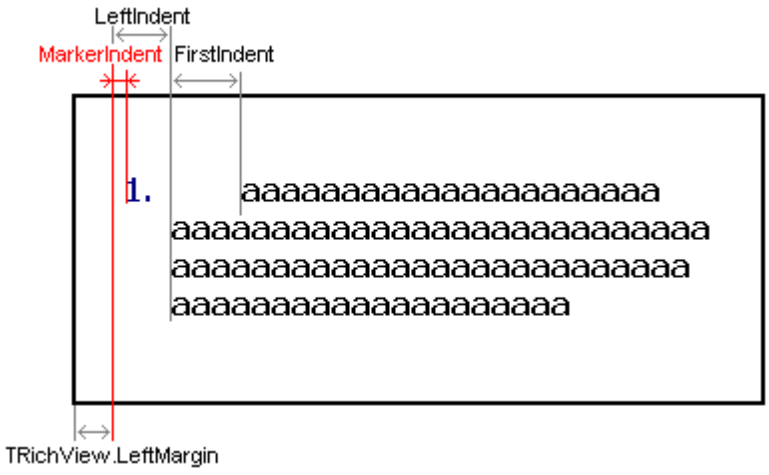
For right-to-left paragraphs [137], list marker position is reversed.

Default value:

rvmaLeft

Indent of the list marker

```
property MarkerIndent: TRVStyleLength[1094];
```



MarkerIndent Example

This value is measured in Units [764] of TRVStyle [739] 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 [866] property.

Default value:

0

See also properties:

- FirstIndent[860];
- LeftIndent[864].

See also properties of TCustomRichView:

- LeftMargin[245].

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: TRVPicture[1037];
```

This property is used if ListType[864]=*rvlstPicture*.

Starting value for marker counter

```
property StartFrom: Integer;
```

For example, if StartFrom for Levels[842][1]=0:

```
1. aaaaa
  0) aaaaa
  1) aaaaa
2. aaaaa
```

Default value:

1

See also:

- ListType[864].

6.2.7.7.2.2 Methods

In TRVListLevel

HasNumbering[868]

Returns *True* if this list level is numbered

```
function HasNumbering: Boolean;
```

This method returns

Result := ListType[864] in [*rvlstDecimal, rvlstLowerAlpha, rvlstUpperAlpha*, *rvlstLowerRoman, rvlstUpperRoman, rvlstImageListCounter*].

6.3 Components ancestor classes

TRichView Components[215] | Ancestor Classes

TRVScroller[869] – ancestor of main visual components: TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[616].

TCustomRVPrint[877] – ancestor of TRVPrint[696] and TRVReportHelper[721]

TCustomRVPrintPreview[888] – of TRVPrintPreview[635]

6.3.1 TRVScroller

This class is an ancestor of all visual components in TRichView family (TRichView[216], TRichViewEdit[469], TDBRichView[604], TDBRichViewEdit[469], TRVPrintPreview[635]).

Unit [VCL/FMX] RVScroll / fmxRVScroll.

Syntax

```
TCustomRVControl = class (TCustomControl)
TRVScroller = class(TCustomRVControl)
```

Hierarchy

Hierarchy (VCL and LCL)

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRVControl

Hierarchy (FireMonkey)

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBox
TCustomRVControl

Main Features

This is a control what can scroll its content.

Do not use this class directly in your program. Use its descendants, such as TRichView[216].

TRVScroller is not registered as a component itself. It introduces some properties and methods used in descendant components.

VCL and LCL: Due to limitation of scrollbar ranges, its vertical scrollbar units are not pixels but special units, "RVSU" (10 pixels by default).

Main Properties and Methods

The most important properties of TRVScroller are:

- VScrollPos[874] – position of vertical scrollbar;
- VScrollMax[874] (read-only) – maximum value of VScrollPos;
- NoHScroll[873] – disallows horizontal scrolling;
- NoVScroll[873] – disallows vertical scrolling;
- HScrollPos[872] – position of horizontal scrollbar, measured in pixels;

- HScrollMax[871] (read-only) – maximum value of HScrollPos;
- InplaceEditor[872] – returns "inplace" editor of some item in document (used in tables for editing cells).

Additional properties for VCL and LCL versions:

- WheelStep[875] defines speed of mouse wheel scrolling.

The most important method is:

- ScrollTo/ScrollToPosition[875], scrolls content of the component vertically to the specified position (in pixels).

6.3.1.1 Properties

In TRVScroller

- BorderStyle[871] [VCL,LCL]
- HScrollMax[871]
- HScrollPos[872]
- HScrollVisible[872] [VCL,LCL]
- InplaceEditor[872]
- NoHScroll[873]
- NoVScroll[873]
- Tracking[873]
- UseXPThemes[873] [VCL,LCL]
- VScrollMax[874]
- VScrollPos[874]
- VScrollVisible[874] [VCL,LCL]
- WheelStep[875] [VCL,LCL]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- Align
- Anchors
- AutoHide [FMX]
- Constraints
- Ctl3D [VCL]
- DisableMouseWheel [FMX]
- DragMode
- EnableDragHighlight [FMX]
- Enabled
- Height
- HelpContext
- HelpKeyword (D6+)
- HelpType (D6+)
- Hint
- Left
- Locked [FMX]
- Margins [FMX]
- Name
- Opacity [FMX]

Padding [FMX]
- ParentCtl3D [VCL]
- ParentShowHint
- PopupMenu

Position [FMX]
RotationAngle [FMX]
RotationCenter [FMX]
Scale [FMX]
ShowHint
ShowScrollBars [FMX]
ShowSizeGrip [FMX]
Size [FMX]
StyleLookup [FMX]
TabOrder
TabStop
- Touch (D2010+) [VCL,FMX]

TouchTargetExpansion [FMX]
- Tag
- Top
- Visible
- Width

6.3.1.1.1 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 [873]

6.3.1.1.2 TRVScroller.HScrollMax

Maximal possible value for HScrollPos [872]

```
property HScrollMax: TRVCoord[1064];
```

See also properties:

- HScrollPos [872];
- VScrollMax [874].

See also:

- Scrolling in RichView [110]

6.3.1.1.3 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: TRVCoord;
```

This value is in range 0..HScrollMax.

See also properties:

- VScrollPos.

See also:

- Scrolling in RichView.

6.3.1.1.4 TRVScroller.HScrollVisible

Set to *False* to hide horizontal scrollbar

```
property HScrollVisible: Boolean;
```

(introduced in version 1.4)

FireMonkey: you can hide/show only the both scrollbars use ShowScrollBars property.

Default value:

True

See also properties:

- VScrollVisible.

See also:

- Scrolling in RichView.

6.3.1.1.5 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 (see "Cell Editing"). In future versions, inplace editors can be used for some other types of items.

Inplace editors are derived from TCustomRichViewEdit 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[489].

6.3.1.1.6 TRVScroller.NoVScroll

Disables vertical and/or horizontal scrolling

```
property NoVScroll: Boolean;
property NoHScroll: Boolean;
```

(introduced in versions 11, 15)

Default value

False

6.3.1.1.7 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

6.3.1.1.8 TRVScroller.UseXPThemes

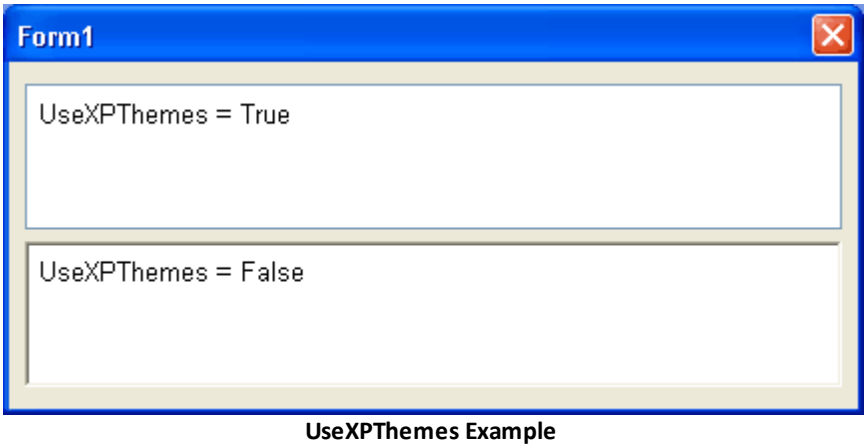
Allows using Windows themes (visual styles)

```
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[871].

6.3.1.1.9 TRVScroller.VScrollMax

Maximal possible value for VScrollPos [874]

```
property VScrollMax: TRVCoord[1064];
```

See also properties:

- VScrollPos [874];
- HScrollMax [871].

See also methods:

- ScrollTo [875].

See also:

- Scrolling in RichView [110].

6.3.1.1.10 TRVScroller.VScrollPos

Position of vertical scrollbar.

```
property VScrollPos: TRVCoord[1064];
```

This value is in range 0..VScrollMax [874].

VCL and LCL:

This value is measured in "RVSU" [110], not in pixels because of Windows limitation on scrollbar scrolling range. You can use this property to scroll document to specified position.

FireMonkey:

This value is measured in logical pixels.

See also properties:

- HScrollPos [872].

See also methods:

- ScrollTo [875].

See also:

- Scrolling in RichView [110].

6.3.1.1.11 TRVScroller.VScrollVisible

Set to *False* to hide vertical scrollbar

```
property VScrollVisible: Boolean;
```

Known problem: scrollbars do not reappear correctly in TRichViewEdit [469] if set VScrollVisible from *False* to *True* when document is not empty.

FireMonkey: you can hide/show only the both scrollbars use ShowScrollBars property.

Default value:

True

See also properties:

- HScrollVisible [872].

See also:

- Scrolling in RichView [110].

6.3.1.1.12 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 [264] *WheelStep*suggested/3) pixels.

Default value:

2

6.3.1.2 Methods

In TRVScroller

Create [875]
ScrollTo [875]

6.3.1.2.1 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 [216];
- TRichViewEdit [469];
- TDBRichView [604];
- TDBRichViewEdit [469];
- TRVPrintPreview [635].

6.3.1.2.2 TRVScroller.ScrollTo[Position]

Scrolls content of the document vertically to the specified position **y** (in pixels).

VCL and LCL:

```
procedure ScrollTo(y: TRVCoord [1064]; SmoothScroll: Boolean = False);
```

(modified in v19)

Vertical scrollbar units are not pixels, so scrolling will be not exactly to specified position (but a little bit higher).

Assign **SmoothScroll** = *True* for a smooth scrolling effect.

FireMonkey:

```
procedure ScrollToPosition(y: TRVCoord [1064]; SmoothScroll: Boolean = False);
```

SmoothScroll is ignored.

See also properties:

- VScrollPos [874];
- VScrollMax [874].

See also:

- Scrolling in TRichView [110].

6.3.1.3 Events

In TRVScroller

- OnHScrolled [877]
- OnVScrolled [877]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

- OnApplyStyleLookup [FMX]
- OnClick
- OnContextPopup [VCL,LCL]
- OnDragDrop
- OnDragEnd [FMX]
- OnDragEnter [FMX]
- OnDragLeave [FMX]
- OnDragOver
- OnEndDrag [VCL,LCL]
- OnEnter
- OnExit
- OnGesture (D2010+) [VCL,FMX]
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseEnter [FMX]
- OnMouseLeave [FMX]
- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown [VCL,LCL]
- OnMouseWheelUp [VCL,LCL]
- OnResize
- OnResized [FMX]
- OnStartDrag [VCL,LCL]

6.3.1.3.1 TRVScroller.OnHScrolled

Occurs when the document is scrolled horizontally

```
property OnHScrolled: TNotifyEvent;
```

See also events:

- OnVScrolled [877].

See also:

- Scrolling in RichView [110].

6.3.1.3.2 TRVScroller.OnVScrolled

Occurs when document is scrolled vertically

```
property OnVScrolled: TNotifyEvent;
```

See also events:

- OnHScrolled [877].

See also properties of TRichView:

- FirstItemVisible [243];
- LastItemVisible [244].

See also:

- Scrolling in RichView [110].

6.3.2 TCustomRVPrint

TCustomRVPrint is an ancestor class of TRVPrint [696] and TRVReportHelper [721] components.

This is an intermediate class. Do not use it directly in your programs.

Unit [VCL/FMX] PtblRV / fmxPtblRV.

Syntax

```
TCustomRVPrint = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

6.3.2.1 Properties

In TCustomRVPrint

- ColorMode [878]
- DarkMode [878]
- EndAt [878]
- FirstPageNo [879]
- IgnorePageBreaks [879]
- LastPageNo [879]
- MaxPrintedItemNo [879]

- MetafileCompatibility[880]
- MetafilePixelsPerInch[880]
- MinPrintedItemNo[881]
- NoMetafiles[881]
- PageNoFromNumber[881]
- PagesCount[882]
- PreviewCorrection[882]
- StartAt[882]
- TransparentBackground[882]

6.3.2.1.1 TCustomRVPrint.ColorMode

Sets color mode for printing and drawing preview

```
property ColorMode: TRVColorMode[1063];
```

(Introduced in version 1.8)

Default values:

- *rvcmColor* for TRVReportHelper[721];
- *rvcmPrinterColor* for TRVPrint[696].

6.3.2.1.2 TCustomRVPrint.DarkMode

Inverts luminance of all colors

```
property DarkMode: Boolean;
```

(Introduced in version 23)

If *True*, luminance of all colors is inverted. Dark colors become light colors, black becomes white, white becomes black.

Be careful with this property. This property is useful when drawing on the screen (using TRVReportHelper[721], or TRVPrint[696] with active VirtualPrinter[707] mode). When printing, black background causes overuse of printer ink.

Default value:

False

See also:

- TCustomRichView[216].DarkMode[234]

6.3.2.1.3 TCustomRVPrint.EndAt

Height of document on the last page, in device pixels.

```
property EndAt: TRVCoord[1064];
```

This property does not include margins.

You can use this property only when document is completely formatted (see FormatPages[711] for TRVPrint, Init[727] and FormatNextPage[726] for TRVReportHelper).

For TRVReportHelper, you can use GetLastPageHeight[726] instead (it's more convenient, because it includes top and bottom margins).

See also properties:

- StartAt [882].

6.3.2.1.4 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 [881] and MaxPrintedItemNo [879] into account. By default, when these properties define the range including the whole document, FirstPageNo returns 1. If MinPrintedItemNo [881]>0, this property returns the index of the page containing the MinPrintedItemNo [881]-th item (see GetPageNo [883]). If MinPrintedItemNo [881]>=ItemCount, FirstPageNo returns PagesCount [882]+1 (and nothing will be printed).

6.3.2.1.5 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 [250][];
- TRVTableItemInfo.Rows[].PageBreakBefore [967].

This property is useful in TRVReportHelper [721] component, if you want to display the loaded document completely in a single rectangle.

Default value

False

6.3.2.1.6 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 [881] and MaxPrintedItemNo [879] into account. By default, when these properties define the range including the whole document, LastPageNo returns PagesCount [882]. If MaxPrintedItemNo [879] is inside the items range (0..ItemCount-1), this property returns the index of the page containing the MaxPrintedItemNo [879]-th item (see GetPageNo [883]).

6.3.2.1.7 TCustomRVPrint.MaxPrintedItemNo

Defines the index of the last item to print.

```
property MaxPrintedItemNo: Integer;
```

(Introduced in version 13)

The component prints items from MinPrintedItemNo [881] 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[879] to LastPageNo[879]).

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

6.3.2.1.8 TCustomRVPrint.MetafileCompatibility

Specifies whether printing output is compatible with metafiles.

```
property MetafileCompatibility: Boolean;
```

(introduced in version 16)

If RVStyle.TextEngine[763] = *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[193]) will print themselves as bitmaps.

Default value

False

6.3.2.1.9 TCustomRVPrint.MetafilePixelsPerInch

Specifies the "pixels per inch" value used when converting metafiles to bitmaps.

```
property MetafilePixelsPerInch: Integer;
```

(introduced in version 17)

Metafiles (in pictures[169] and *hot-pictures*[172]) are printed as bitmaps, if NoMetafiles[881] = *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).

6.3.2.1.10 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[879]. 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[879] to LastPageNo[879]).

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

6.3.2.1.11 TCustomRVPrint.NoMetafiles

Allows drawing metafiles as bitmaps.

```
property NoMetafiles: Boolean;
```

(introduced in version 15)

If *True*, metafiles (in pictures[169] and *hot-pictures*[172]) and equations[193] 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[880]

6.3.2.1.12 TCustomRVPrint.PageNoFromNumber

Specifies the number of the first page for "page number" fields[190].

```
property PageNoFromNumber: Integer;
```

(introduced in version 15)

Default value:

1

See also:

- TRVReportHelper.OnGetPageNumber[728]

6.3.2.1.13 TCustomRVPrint.PagesCount

Total number of pages in the document.

```
property PagesCount: Integer;
```

You can use this property only when document is formatted (see FormatPages[711] for TRVPrint, Init[727] and FormatNextPage[726] for TRVReportHelper).

6.3.2.1.14 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

6.3.2.1.15 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: TRVCoord[1064];
```

This property is useful (in conjunction with EndAt[878]) in TRVPrint.ContinuousPrint[709] method.

Default value:

0

6.3.2.1.16 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[919] of tables.

6.3.2.2 Methods

In TCustomRVPrint

CanSavePDF[883]
Clear[883]
GetFirstItemOnPage[883]

GetPageNo 883
UpdatePaletteInfo 884

6.3.2.2.1 TCustomRVPrint.CanSavePDF

Returns True if PDF saving is available.

```
function CanSavePDF: Boolean;
```

If this method returns *True*, you can use TRVPrint.SavePDF 715 and TRVReportHelper.SavePDF methods.

Currently, PDF saving is implemented only in FireMonkey, if Skia4Delphi 163 is available.

6.3.2.2.2 TCustomRVPrint.Clear

Clears memory allocated for printing (in FormatPages 711)

```
procedure Clear;
```

After calling Clear, TRVPrint will not be able to print and display preview until the next call of AssignSource 709 and FormatPages 711 (or Init 727 and FormatNextPage 726 in TRVReportHelper).

6.3.2.2.3 TCustomRVPrint.GetFirstItemOnPage

Returns index of the first item on the page

```
procedure GetFirstItemOnPage(PageNo: Integer;
  out 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 901. Tables can be printed on several pages, and using this information you cannot determine which part of table starts the page.

See also:

- GetPageNo 883.

See also properties:

- PagesCount 882.

6.3.2.2.4 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[254] of the source TRichView control, or table cell, or RVData[254] of a cell inplace editor.

For TRVPrint[696], the source TRichView control is a control assigned by AssignSource[709]. For TRVReportHelper, the source TRichView control is RichView[724] 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[167], 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[883].

See also methods of TCustomRichView:

- GetOffsAfterItem[316];
- GetOffsBeforeItem[317].

6.3.2.2.5 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[242]

TRVPrint uses the same method of working in 256 color mode as its source RichView (see RichView.DoInPaletteMode[241]). Information about source RichView.DoInPaletteMode is updated when executing AssignSource[709] 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[635].

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[893] or RVPrint[638] properties.

See also methods of TRichView:

- UpdatePaletteInfo[375].

6.3.2.3 Events

In TCustomRVPrint

- OnAfterPrintImage [885]
- OnDrawCheckpoint [885]
- OnDrawHyperlink [886]
- OnPrintComponent [887]

6.3.2.3.1 TCustomRVPrint.OnAfterPrintImage

Allows custom printing on pictures [169] and hot-pictures [172].

VCL and LCL:

```
type
  TRVAfterPrintImageEvent = procedure(Sender: TCustomRVPrint[877];
    ARVData: TCustomRVData[1014]; AItem: TCustomRVItemInfo;
    ACanvas: TRVCanvas;
    const ARect: TRVCoordRect[1064]; APSaD: PRVScreenAndDevice[1090]) of object;
```

FireMonkey:

```
type
  TRVAfterPrintImageEvent = procedure(Sender: TCustomRVPrint[877];
    ARVData: TCustomRVData[1014]; AItem: TCustomRVItemInfo;
    ACanvas: TRVFMXCanvas[1018];
    const ARect: TRVCoordRect[1064]; APSaD: PRVScreenAndDevice[1090]) of object;
```

```
property OnAfterPrintImage: TRVAfterPrintImageEvent;
```

(introduced in version 20)

Parameters:

ACanvas – canvas for printing.

ARVData – a document containing the item.

AItem – an item that is being drawn.

ARect – image rectangle on **ACanvas**; it does not include spacing and borders.

This event occurs when the image is already printed. It allows printing additional content on top of it.

See also:

- TRichView.OnAfterDrawImage [378]

6.3.2.3.2 TCustomRVPrint.OnDrawCheckpoint

Occurs when the component draws (or prints) an item having a *checkpoint* [92]

```
type
  TRVDrawCheckpointEvent = procedure (Sender: TCustomRVPrint[877];
```

```
    RVData: TCustomRVData[1014]; ItemNo: Integer; X, Y: TRVCoord[1064])
    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[886].

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).

6.3.2.3.3 TCustomRVPrint.OnDrawHyperlink

Occurs when the component draws a hyperlink

```
type
  TRVDrawHyperlinkEvent =
    procedure (Sender: TCustomRVPrint[877];
      RVData: TCustomRVData[1014]; ItemNo: Integer;
      R: TRVCoordRect[1064]) 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[885].

6.3.2.3.4 TCustomRVPrint.OnPrintComponent

Occurs when printing inserted control [174]

```
property OnPrintComponent: TRVPrintComponentEvent;

TRVPrintComponentEvent = procedure(Sender: TCustomRVPrint[877];
  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 [1043]. 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[1043](PrintMe)
end;
```

In TRVPrint, this event is generated by:

- Print[713];
- PrintPages[714];
- MakePreview[712];
- MakeScaledPreview[713];
- DrawPreview[711];
- when drawing TRVPrintPreview[635] component.

6.3.3 TCustomRVPrintPreview

TCustomRVPrintPreview is an ancestor of TRVPrintPreview[635] component. This class draws a blank page, introduces mechanisms of page switching and scaling.

Unit [VCL/FMX] CRVPP / fmxCRVPP.

Syntax

```
TCustomRVPrintPreview = class(TRVScroller[869])
```

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[869]

Cursors

Print preview uses two cursors:

- ZoomInCursor[895],
- ZoomOutCursor[896].

Scaling

Scaling is defined by:

- ZoomMode[895] property – mode of scaling;
- ZoomPercent[896] property – scaling percent;

- SetZoom[898] method changes ZoomPercent[896];
- OnZoomChanged[899] event occurs when ZoomPercent[896] changes;
- ClickMode[891] property allows/disallows zooming.

Page switching

First[897], Last[897], Prev[898], Next[897] methods, PageNo[893] property.

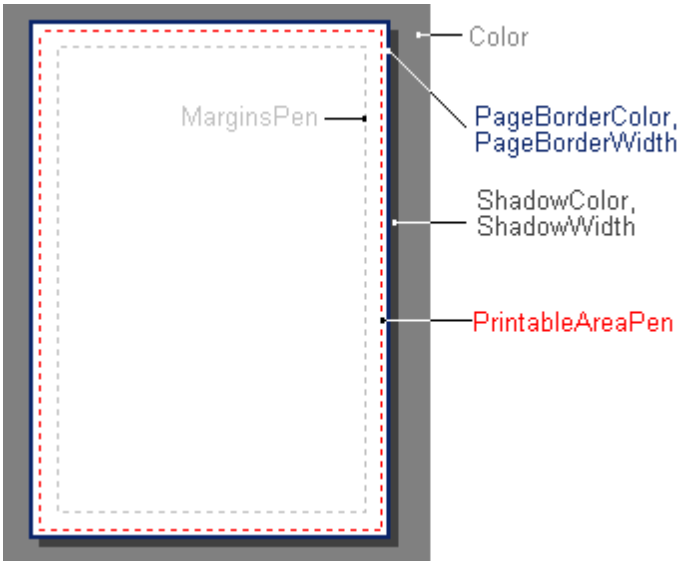
Displaying margins

MarginsPen/MarginsStroke property[892]

Colors and widths

- Color/Fill[891] – background fill;
- PageBorderColor[893], PageBorderWidth[893] define border around the page;
- ShadowColor[894], ShadowWidth[894] define shadow;
- DarkModeUI[892] inverts luminance of colors in user interface of this control.

Scheme



TCustomRVPrintPreview

The scheme above is for VCL and LCL version of the component.

FireMonkey version uses Fill[891] property instead of Color property.

6.3.3.1 Properties

In TCustomRVPrintPreview

- ClickMode[891]
- Color[891] [VCL,LCL]
- Fill[891] [FMX]
- MarginsPen[892] [VCL, LCL]
- MarginsStroke[892] [FMX]
- PageBorderColor[893]

PageBorderWidth 893
PageNo 893
PrintableAreaPen 893 [VCL, LCL]
PrintableAreaStroke 893 [FMX]
ShadowColor 894
ShadowWidth 894
ZoomInCursor 895
ZoomMode 895
ZoomOutCursor 896
ZoomPercent 896

Derived from TRVScroller 869

- BorderStyle 871 [VCL]
- HScrollMax 871
- HScrollPos 872
- HScrollVisible 872 [VCL,LCL]
- InplaceEditor 872 (not used in this component)
- NoVScroll 873
- Tracking 873
- UseXPThemes 873 [VCL,LCL]
- VScrollMax 874
- VScrollPos 874
- VScrollVisible 874 [VCL,LCL]
- WheelStep 875

Derived from TCustomControl [VCL/LCL] or TCustomScrollBox [FMX]

Align
Anchors
AutoHide [FMX]
Constraints
Ctl3D [VCL]
DisableMouseWheel [FMX]
DragMode
EnableDragHighlight [FMX]
Enabled
Height
HelpContext
HelpKeyword (D6+)
HelpType (D6+)
Hint
Left
Locked [FMX]
Margins [FMX]
Name
Opacity [FMX]
Padding [FMX]

- ParentCtl3D [VCL]
- ParentShowHint
- PopupMenu
- Position [FMX]
- RotationAngle [FMX]
- RotationCenter [FMX]
- Scale [FMX]
- ShowHint
- ShowScrollBars [FMX]
- ShowSizeGrip [FMX]
- Size [FMX]
- StyleLookup [FMX]
- TabOrder
- TabStop
- Touch (D2010+) [VCL,FMX]
- TouchTargetExpansion [FMX]
- Tag
- Top
- Visible
- Width

6.3.3.1.1 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 [895] and zoom-out [896] cursors are not used.
rvcmSwitchZoom	Clicking switches between 100% and custom zoom.

Default value:

rvcmSwitchZoom

6.3.3.1.2 TCustomRVPrintPreview.Color, Fill

Background fill.

VCL:

```
property Color: TRVColor[1063];
```

(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.

FireMonkey:

```
property Fill: TBrush;
```

(introduced in v20)

Default value:

rvclBtnShadow[1105]

See also:

- DarkModeUI[892]
- Picture at the bottom of TCustomRVPrintPreview[888] topic.

6.3.3.1.3 TCustomRVPrintPreview.DarkModeUI

Inverts luminance of all colors in user interface.

```
property DarkModeUI: Boolean;
```

(introduced in version 23)

Dark colors become light colors, black becomes white, white becomes black.

This property affects all colors used in user interface, including background fill, page border and shadow, etc.

It does not affect drawing on pages (for pages, RVPrint[638].DarkMode[878] is used)

Default value

False

6.3.3.1.4 TCustomRVPrintPreview.MarginsPen, MarginsStroke

This pen is used for drawing rectangle around the document area (i.e. inside page margins) on the page preview.

VCL:

```
type
  TRVMarginsPen = class (TPen)
    property Style default psClear;
    property Color default clSilver;
  end;

property MarginsPen: TRVMarginsPen;
```

By default, MarginsPen.Style=*psClear*, so the rectangle is invisible.

FMX:

```
type
  TRVMarginsPen = class (TStrokeBrush);

property MarginsStroke: TRVMarginsPen;
```

By default, MarginsStroke.Kind=*TBrushKind.None*, so the rectangle is invisible.

See also properties:

- PrintableAreaPen/PrintableAreaStroke[893].

See also properties of TRVPrint[696]:

- Margins[705],

See also:

- Picture in TRVPrint topic[696].

6.3.3.1.5 TCustomRVPrintPreview.PageBorderColor

Color of frame around the page.

```
property PageBorderColor: TRVColor[1063];
```

(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:

rvclHighlight[1105]

See also:

- DarkModeUI[892]
- Picture at the bottom of TCustomRVPrintPreview[888] topic.

6.3.3.1.6 TCustomRVPrintPreview.PageBorderWidth

Width of frame around the page, pixels

```
property PageBorderWidth: TRVPixel96Length[1085];
```

(introduced in v1.8)

Default value:

2

See also:

- DarkModeUI[892]
- Picture at the bottom of TCustomRVPrintPreview[888] topic.

6.3.3.1.7 TCustomRVPrintPreview.PageNo

Index of displayed page (from 1)

```
property PageNo: Integer;
```

See also methods:

- Prev[898];
- Next[897];
- First[897];
- Last[897].

6.3.3.1.8 TCustomRVPrintPreview.PrintableAreaPen, PrintableAreaStroke

This pen is used for drawing rectangle around the printable area on the page preview.

VCL:

```
type
  TRVPAPen = class (TPen)
    property Style default psClear;
    property Color default clRed;
  end;

property PrintableAreaPen: TRVPAPen;
```

(introduced in version 10)

By default, PrintableAreaPen.Style=*psClear*, so the rectangle is invisible.

FMX:

```
type
  TRVAPen = class (TStrokeBrush);

property PrintableAreaPen: TRVAPen;
```

(introduced in version 20)

By default, PrintableAreaStroke.Kind=*TBrushKind.None*, so the rectangle is invisible.

Printer cannot print beyond the area shown on this rectangle.

Windows: see WinAPI function GetDeviceCaps, PHYSICAL*** constants.

See also properties:

- MarginsPen/MarginsStroke [892].

6.3.3.1.9 TCustomRVPrintPreview.ShadowColor

Color of the page shadow.

```
property ShadowColor: TRVColor[1063];
```

(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:

rvcl3DDkShadow[1105]

See also:

- DarkModeUI [892]
- Picture at the bottom of TCustomRVPrintPreview [888] topic.

6.3.3.1.10 TCustomRVPrintPreview.ShadowWidth

Width of the page shadow, pixels.

```
property ShadowWidth: TRVPixel96Length[1085];
```

(introduced in v1.8)

Default value:

20

See also:

- Picture at the bottom of TCustomRVPrintPreview[888] topic.

6.3.3.1.11 TCustomRVPrintPreview.ZoomInCursor

Cursor appeared when the current scale<100%, or the current scale=100% switched from the scale>100%

```
property ZoomInCursor: TCursor;
```

If ZoomPercent[896]<>100%, clicking on component changes ZoomPercent[896] to 100%.

If ZoomPercent[896]=100%, clicking on component changes ZoomPercent[896] to previous scale.

Default value:

crRVZoomIn (102)

(this cursor is registered automatically)

See also properties:

- ZoomOutCursor[896];
- ZoomPercent[896].

See also:

- RichView cursors[147].

6.3.3.1.12 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[896] to display full page, keeping the full-page view during resizing (changing ZoomPercent[896] when needed)
rvzmPageWidth	Changing ZoomPercent[896] to fit width of page, keeping the page-width view during resizing (changing ZoomPercent[896] when needed)
rvzmCustom	Scaling according to the ZoomPercent[896] value

Default value:

rvzmCustom

6.3.3.1.13 TCustomRVPrintPreview.ZoomOutCursor

Cursor appeared when the current scale>100%, or the current scale=100% switched from the scale<100%

```
property ZoomOutCursor: TCursor;
```

If ZoomPercent[896]<>100%, clicking on component changes ZoomPercent[896] to 100%.

If ZoomPercent[896]=100%, clicking on component changes ZoomPercent[896] to previous scale.

Default value:

crRVZoomOut (103)

(this cursor is registered automatically)

See also properties:

- ZoomInCursor[895];
- ZoomPercent[896].

See also:

- RichView cursors[147].

6.3.3.1.14 TCustomRVPrintPreview.ZoomPercent

Scale of preview, %

```
property ZoomPercent: TRVZoomPercent[1102];
```

Although assigning to this property is possible, the preferable way to change preview scale is calling SetZoom[898] method.

This property is changed automatically when resizing the component, if ZoomMode[895] = *rvzmFullPage* or *rvzmPageWidth*.

It's recommended to keep this property in 10%–500% range.

See also methods:

- SetZoom[898].

See also properties:

- ZoomMode[895].

See also events:

- OnZoomChanged[899].

6.3.3.2 Methods

In TCustomRVPrintPreview

Create[897]
First[897]
Last[897]
Next[897]
Prev[898]
SetZoom[898]

Derived from TRVScroller [869]

ScrollTo [875] (not used in this component)

6.3.3.2.1 TCustomRVPrintPreview.Create

A constructor, creates a new TCustomRVPrintPreview object.

```
constructor Create(AOwner: TComponent);override;
```

Do not to create TCustomRVPrintPreview objects, create TRVPrintPreview [635] components instead.

6.3.3.2.2 TCustomRVPrintPreview.First

Switches the current page (PageNo [893] property) to the first page.

```
procedure First;
```

See also properties:

- PageNo [893].

See also methods:

- Prev [898];
- Next [897];
- Last [897].

6.3.3.2.3 TCustomRVPrintPreview.Last

Switches the current page (PageNo [893] property) to the last page.

```
procedure Last;
```

See also properties:

- PageNo [893];
- TRVPrintPreview.RVPrint [638].PagesCount [882].

See also methods:

- Prev [898];
- Next [897];
- First [897].

6.3.3.2.4 TCustomRVPrintPreview.Next

Switches the current page (PageNo [893] property) to the next page, if possible.

```
procedure Next;
```

See also properties:

- PageNo [893];
- TRVPrintPreview.RVPrint [638].PagesCount [882].

See also methods:

- Prev [898];
- First [897];
- Last [897].

6.3.3.2.5 TCustomRVPrintPreview.Prev

Switches the current page (PageNo[893] property) to the previous page, if possible.

```
procedure Prev;
```

See also properties:

- PageNo[893].

See also methods:

- Next[897];
- First[897];
- Last[897].

6.3.3.2.6 TCustomRVPrintPreview.SetZoom

Changes ZoomMode[895] to *rvzmCustom* and ZoomPercent[896] to **Percent**.

```
procedure SetZoom(Percent: Integer[1102]);
```

Calling this method has an immediate effect.

It's recommended to keep **Percent** in 10%–500% range.

See also properties:

- ZoomMode[895];
- ZoomPercent[896].

6.3.3.3 Events

In TCustomRVPrintPreview

- OnZoomChanged[899]

Derived from TRVScroller[869]

- OnHScrolled[877]
- OnVScrolled[877]

Derived from TCustomControl

- OnClick
- OnContextPopup
- OnDragDrop
- OnDragOver
- OnEndDrag
- OnEnter
- OnExit
- OnGesture (D2010+)
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseMove
- OnMouseWheel
- OnMouseWheelDown
- OnMouseWheelUp

- OnResize
- OnStartDrag

6.3.3.3.1 TCustomRVPrintPreview.OnZoomChanged

Occurs when ZoomPercent[896] changes.

```
property OnZoomChanged: TNotifyEvent;
```

7 Item types

TCustomRVItemInfo is a basic type for all items[163] in TRichView documents.

A 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[216] and TRichViewEdit[469] instead of working with the item classes directly, so details of implementation are hidden from programmers. The exceptions are:

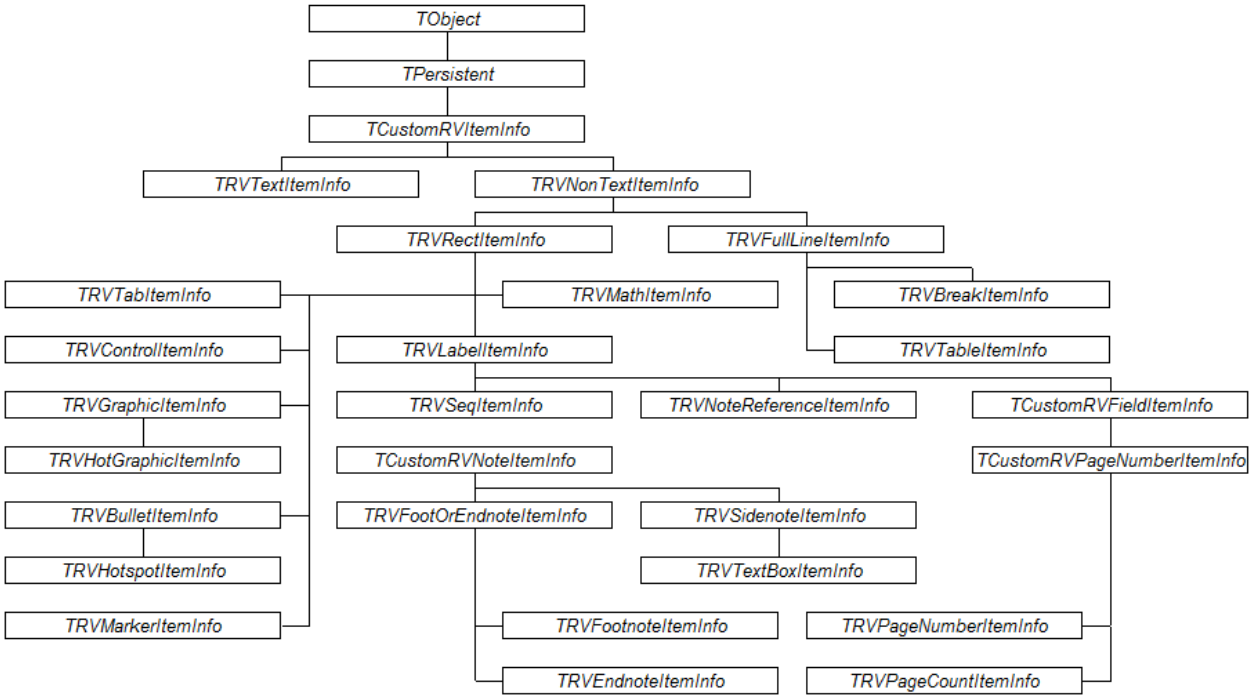
- TRVTableItemInfo[901];
- TRVLabelItemInfo[970] and classes, inherited from it.

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[167].
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[168].
TRVControlItemInfo	RVItem	Item containing Delphi control[174].
TRVGraphiItemInfo	RVItem	Picture[169].
TRVHotGraphicItemInfo	RVItem	*Hot-picture*[172] (picture-hyperlink).
TRVBulletItemInfo	RVItem	*Bullet*[176] (image from TImageList).
TRVHotspotItemInfo	RVItem	*Hotspot*[178] (*bullet*-hyperlink).
TRVMarkerItemInfo	RVMarker	Paragraph marker[180].
TRVTableItemInfo[901]	RVTable	Table[179].
TRVBreakItemInfo	RVItem	*Break*[173] (horizontal line).

Class	Unit	Meaning
TRVLabelItemInfo 970	RVLabelItem	Label 181 (non-text item looking like text).
TRVSeqItemInfo 978	RVSeqItem	Numbered sequence 183.
TCustomRVNoteItemInfo 982	RVNote	Basic class for notes and text boxes.
TRVFootOrEndnoteItemInfo 982	RVNote	Basic class for footnotes and endnotes
TRVEndnoteItemInfo 985	RVNote	Endnote 184.
TRVFootnoteItemInfo 988	RVNote	Footnote 185.
TRVSidenoteItemInfo 990	RVSidenote	Sidenote 186
TRVTextBoxItemInfo 1004	RVSidenote	Text box 188
TRVNoteReferenceItemInfo 1006	RVNote	Reference to the parent note 190
TRVPageNumberItemInfo 1010	RVFieldItems	"Page number" field 190
TRVPageCountItemInfo 1011	RVFieldItems	"Page count" field 192
TRVMathItemInfo 974	RVMathItem	Mathematical expression 193

Hierarchy



Other Types

```
TCustomRVItemInfoClass = class of TCustomRVItemInfoClass;
```

7.1 Table - TRVTableItemInfo

TRVTableItemInfo is a class representing table [179] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem [276]) or inserted into TRichViewEdit (see InsertItem [531]). Style [310] of this item type: *rvsTable* (-60)

Unit RVTable.

Syntax

```
TRVTableItemInfo = class (TRVFullLineItemInfo)
```

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]
TRVNonTextItemInfo
TRVFullLineItemInfo

Overview

See overview topics about tables [196].

Properties

The most of properties of this class are published. These properties can be saved/loaded in RVF [129] files or streams.

The main property of TRVTableItemInfo is Cells [912]. The number of rows is RowCount [920], number of columns is ColCount [914].

Properties of rows are available via Rows [920] property.

Width of table can be defined using BestWidth [906] property.

Properties defining border around the table:

- BorderWidth [909];
- BorderStyle [908];
- BorderColor [906];
- BorderLightColor [907];
- VisibleBorders [921];
- also BorderHSpacing [907], BorderVSpacing [908].

Properties defining default border around cells:

- CellBorderWidth [910];
- CellBorderStyle [910];
- CellBorderColor [909];
- CellBorderLightColor [910].

Properties defining default colors of cells in rows and columns:

- HeadingRowColor[923], LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor

Properties defining spacing between cells:

- CellHSpacing[911];
- CellVSpacing[913];
- BorderHSpacing[907];
- BorderVSpacing[908];
- also CellHPadding[911], CellVPadding[913].

Background properties:

- Color[914];
- Opacity[916];
- BackgroundImage[904];
- BackgroundProperties[905];
- BackgroundImageFileName[905].

Border colors:

- BorderColor[906];
- BorderLightColor[907];
- CellBorderColor[909];
- CellBorderLightColor[910].

Rules:

- HRuleWidth[916];
- HRuleColor[915];
- HOutermostRule[915];
- VRuleWidth[922];
- VRuleColor[922];
- VOutermostRule[921].

Export:

- TextColSeparator[920];
- TextRowSeparator[921];
- BackgroundImageFileName[905].

Other properties:

- Options[916];
- PrintOptions[919];
- HeadingRowCount[914].

Methods

See operations on tables[205].

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[129] files or streams. Reassign them when tables are loaded, use TRichView.OnItemAction[387] event.

Events:

- OnCellEditing[948];
- OnDrawBorder[949].

Related

- Drawing table grid[750];
- New row auto insertion[1117].

7.1.1 Properties

In TRVTableItemInfo

- BackgroundImage[904]
- BackgroundImageFileName[905]
- BackgroundProperties[905]
- BestWidth[906]
- BorderColor[906]
- BorderLightColor[907]
- BorderHSpacing[907]
- BorderVSpacing[908]
- BorderStyle[908]
- BorderWidth[909]
- CellBorderColor[909]
- CellBorderLightColor[910]
- CellBorderStyle[910]
- CellBorderWidth[910]
- CellHPadding[911]
- CellHSpacing[911]
- CellOverrideColor[912]
- CellPadding[912]
- Cells[912]
- CellVPadding[913]
- CellVSpacing[913]
- ColCount[914]
- Color[914]
- EvenColumnsColor[923]
- EvenRowsColor[923]
- FirstColumnColor[923]
- HeadingRowColor[923]
- HeadingRowCount[914]
- HOutermostRule[915]
- HRuleColor[915]
- HRuleWidth[916]
- LastColumnColor[923]
- LastRowColor[923]

- OddColumnsColor 923
- OddRowsColor 923
- Opacity 916
- Options 916
- PrintOptions 919
- RowCount 920
- Rows 920
- TextColSeparator 920
- TextRowSeparator 921
- VisibleBorders 921
- VOutermostRule 921
- VRuleColor 922
- VRuleWidth 922

7.1.1.1 TRVTableItemInfo.BackgroundImage

Background image for the table.

```
property BackgroundImage: TRVGraphic 1032;
```

(introduced in v1.8)

Visibility, size, position and repetition of the image is defined in the BackgroundProperties 905 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 1114).

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:

- BackgroundProperties 905;
- Color 914.

See also properties of table cell:

- BackgroundImage 953;
- BackgroundProperties 954.

See also properties of TRichView:

- BackgroundPicture[229];
- BackgroundProperties[229].

7.1.1.2 TRVTableItemInfo.BackgroundImageFileName

A string storing the file name associated with BackgroundImage[904].

```
property BackgroundImageFileName: TRVUnicodeString[1099];
```

(introduced in v1.9; changed in version 18)

If *rvhtmlsioUseItemImageFileNames* in HTMLSaveProperties[244].ImageOptions[442], 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[1072] *rvespImageFileName*.

Assignment to this property cannot be undone by the user[120]; to make an undoable assignment, use *rvespImageFileName* property.

See also properties of table cell:

- BackgroundImageFileName[953].

7.1.1.3 TRVTableItemInfo.BackgroundStyle

Defines position of BackgroundImage[904].

```
property BackgroundStyle: TRVItemBackgroundStyle[1080];
```

(introduced in v1.8)

Deprecated, use BackgroundProperties[905] instead.

See also:

- Undo of assignments to table properties.

See also properties:

- BackgroundImage[904];
- Color[914].

See also properties of table cell:

- BackgroundProperties[954];
- BackgroundImage[953].

See also properties of TRichView:

- BackgroundProperties[229];
- BackgroundPicture[229].

7.1.1.4 TRVTableItemInfo.BackgroundProperties

Defines visibility, size, position, and repetition of BackgroundImage[904].

```
property BackgroundProperties: TRVBackgroundProperties[1021];
```

(introduced in v24)

See also:

- Undo of assignments to table properties.

See also properties:

- BackgroundImage[904];
- Color[914].

See also methods:

- Color[940].

See also properties of table cell:

- BackgroundProperties[954];
- BackgroundImage[953].

See also properties of TRichView:

- BackgroundProperties[229];
- BackgroundPicture[229].

7.1.1.5 TRVTableItemInfo.BestWidth

Preferred width for the table.

```
property BestWidth: TRVHTMLLength[1079];
```

This is a preferred width for the table, measured either in Units[764] of the linked TRVStyle[739] component 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), TRichView tries to calculate the table width basing on its cells. In this version TRichView can do it only if all columns have widths defined in TRVStyleUnits[1094] (have at least one cell with BestWidth[955] defined in TRVStyleUnits[1094], and do not have cells with BestWidth[955] 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[211];
- Undo of assignments to table properties.

See also properties of table cell:

- BestWidth[955];
- BestHeight[955].

7.1.1.6 TRVTableItemInfo.BorderColor

Defines color of the table border.

```
property BorderColor: TRVColor[1063]
```

This property defines either color of flat border or "dark" color of 3d border around the table.

See BorderStyle[908] for details.

This property corresponds to HTML <table bordercolor> or <table bordercolordark> attributes (specific to Microsoft Internet Explorer).

Default value:

rvclWindowText[1105]

See also:

- Undo of assignments to table properties;
- Colors and layout of tables[199] (with pictures).

See also properties

- BorderLightColor[907];
- CellBorderColor[909];
- CellBorderLightColor[910].

7.1.1.7 TRVTableItemInfo.BorderHSpacing

Horizontal spacing between the table border and the outermost cells.

```
property BorderHSpacing: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] component.

BorderHSpacing be saved in HTML+CSS if:

- BorderHSpacing=CellHSpacing[911], or
- BorderHSpacing>=CellHSpacing[911] and CellHSpacing[911]>=0

Default value:

2

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- BorderVSpacing[908];
- CellHSpacing[911];
- CellVSpacing[913].

7.1.1.8 TRVTableItemInfo.BorderLightColor

This property defines "light" color of 3d border around the table.

```
property BorderLightColor: TRVColor[1063];
```

See BorderStyle[908] for details.

This property corresponds to HTML <table bordercolorlight> attribute (specific to Microsoft Internet Explorer).

Default value:

rvclBtnHighlight[1105]

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [199] (with pictures).

See also properties:

- BorderColor [906];
- CellBorderColor [909];
- CellBorderLightColor [910].

7.1.1.9 TRVTableItemInfo.BorderStyle

This property defines style of border around the table.

```
property BorderStyle: TRVTableBorderStyle[1096];
```

If you want to hide table border, set BorderWidth [909] = 0.

Default value:

rvtbRaised

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [199] (with pictures).

See also properties:

- CellBorderStyle [910];
- BorderColor [906];
- BorderLightColor [907];
- BorderWidth [909];
- VisibleBorders [921].

7.1.1.10 TRVTableItemInfo.BorderVSpacing

Vertical spacing between border and the outermost cells.

```
property BorderVSpacing: TRVStyleLength[1094];
```

This value is measured in Units [764] of the linked TRVStyle [739] component.

BorderVSpacing be saved in HTML+CSS if:

- BorderVSpacing=CellVSpacing [913], or
- BorderVSpacing>=CellVSpacing [913] and CellVSpacing [913]>=0

Default value:

2

Note: this default value assumes that RVStyle.Units [764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [199] (with pictures).

See also properties:

- BorderHSpacing [907];
- CellHSpacing [911];
- CellVSpacing [913].

7.1.1.11 TRVTableItemInfo.BorderWidth

Width of border around the table.

```
property BorderWidth: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] 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[199] (with pictures).

See also properties:

- CellBorderWidth[910];
- VisibleBorders[921].

7.1.1.12 TRVTableItemInfo.CellBorderColor

Defines default border color for all table cells.

```
property CellBorderColor: TRVColor[1063]
```

This property defines either default color of flat border or default "dark" color of 3d border around the table cells.

See CellBorderStyle[910] for details.

This value can be overridden with BorderColor[956] of cell.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

Default value:

rvclWindowText[1105]

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- CellBorderLightColor[910];
- BorderLightColor[907];
- BorderColor[906].

See also properties of table cell:

- BorderColor[956];
- BorderLightColor[956].

7.1.1.13 TRVTableItemInfo.CellBorderLightColor

Defines default "light" color of 3d border for all table cells.

```
property CellBorderLightColor: TRVColor[1063];
```

See CellBorderStyle[910] for details. To hide cell borders, set CellBorderWidth[910] = 0.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

This value can be overridden with BorderLightColor[956] of cell.

Default value:

rvclBtnHighlight[1105]

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- CellBorderColor[909];
- BorderLightColor[907];
- BorderColor[906].

See also properties of table cell:

- BorderColor[956];
- BorderLightColor[956].

7.1.1.14 TRVTableItemInfo.CellBorderStyle

Style of border around the table cells

```
property BorderStyle: TRVTableBorderStyle[1096];
```

To hide cell borders, set CellBorderWidth[910] = 0.

Default value:

rvtbLowered

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- BorderStyle[908];
- CellBorderWidth[910];
- CellBorderColor[909];
- CellBorderLightColor[910].

7.1.1.15 TRVTableItemInfo.CellBorderWidth

Width of border around the cells of table, pixels

```
property CellBorderWidth: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] component.

Set to 0 to hide border.

If value of this property <> BorderWidth[909], 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[199] (with pictures).

See also properties:

- BorderWidth[909];
- CellBorderStyle[910];
- CellBorderColor[909];
- CellBorderLightColor[910].

7.1.1.16 TRVTableItemInfo.CellHPadding

Horizontal distance from the cell content to the cell border.

```
property CellHPadding: TRVStyleLength[1094];
```

(introduced in v10)

This value is measured in Units[764] of the linked TRVStyle[739] component.

This value affects all cells in the table.

If the cell is rotated[961], CellHPadding and CellVPadding[913] are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- CellVPadding[913];
- CellPadding[912].

7.1.1.17 TRVTableItemInfo.CellHSpacing

Horizontal spacing between cells.

```
property CellHSpacing: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] component.

This value can be negative. For example, you can assign CellHSpacing=-CellBorderWidth[910] to collapse cell borders horizontally.

Default value:

2

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [199] (with pictures).

See also properties:

- CellVSpacing [913];
- BorderHSpacing [907];
- BorderVSpacing [908].

7.1.1.18 TRVTableItemInfo.CellOverrideColor

A color to override cell background colors.

```
property CellOverrideColor: TRVColor[1063];
```

(introduced in v17)

If **CellOverrideColor**<>*rvclNone* [1105], all cells are drawn using this color. Colors of individual cells [957] and colors of bands of rows and columns [923] are ignored.

Default value:

rvclNone [1105]

7.1.1.19 TRVTableItemInfo.CellPadding

Distance between cell content and cell border.

```
property CellPadding: TRVStyleLength[1094];
```

This property is maintained for backward compatibility. Assigning to this property assigns values to CellHPadding [911] and CellVPadding [913]. Reading this property returns Round((CellHPadding [911] +CellVPadding [913])/2).

7.1.1.20 TRVTableItemInfo.Cells

Provides indexed access to all cells in the table.

```
property Cells[Row, Col: Integer]: TRVTableCellData[951];
```

This is the key property of table.

Row can be in range from 0 to RowCount [920]-1. **Col** can be in range from 0 to ColCount [914]-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 [920].

See also:

- Table Cells Overview [198].

7.1.1.21 TRVTableItemInfo.CellVPadding

Vertical distance from the cell content to the cell border.

```
property CellVPadding: TRVStyleLength[1094];
```

(introduced in v10)

This value is measured in Units[764] of the linked TRVStyle[739] component.

This value affects all cells in the table.

If the cell is rotated[961], CellHPadding[911] and CellVPadding are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- CellHPadding[911];
- CellPadding[912].

7.1.1.22 TRVTableItemInfo.CellVSpacing

Vertical spacing between cells.

```
property CellVSpacing: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] component.

This value can be negative. For example, you can assign CellVSpacing=-CellBorderWidth[910] to collapse cell borders vertically.

Default value:

2

Note: this default value assumes that RVStyle.Units[764]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[199] (with pictures).

See also properties:

- CellHSpacing[911];
- BorderHSpacing[907];
- BorderVSpacing[908].

7.1.1.23 TRVTableItemInfo.ColBandSize, RowBandSize

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 [923].

The heading (or the first row) and the first column may be excluded from bands.

Default values

1

7.1.1.24 TRVTableItemInfo.ColCount

Returns the number of columns in the table.

```
property ColCount: Integer
```

(introduced in v10)

If RowCount [920]>0, this property returns Rows [920][0].Count (all rows have the same number of cells). If RowCount [920]=0, this property returns 0.

See also properties:

- Cells [912];
- Rows [920].

7.1.1.25 TRVTableItemInfo.Color

Background color of the table

```
property Color: TRVColor[1063];
```

Set to *rvclNone* [1105] for transparent background. You can specify the background Opacity [916].

If the table has a background image [904], this image is drawn on top of color filling.

Default value:

rvclWindow [1105]

See also:

- Undo of assignments to table properties;
- Colors and layout of tables [199] (with pictures).

7.1.1.26 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 [923]) 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 [919],
- ColBandSize, RowBandSize [913]

7.1.1.27 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 [916]>0)

Note about rules.

Default value:

False

See also:
- Undo of assignments to table properties;
- Table rules [200] (with pictures).

See also properties:
- VOutermostRule [921];
- HRuleWidth [916];
- HRuleColor [915].

7.1.1.28 TRVTableItemInfo.HRuleColor

Specifies the color of horizontal rules (horizontal lines between cells)

```
property HRuleColor: TRVColor [1063];
```

Horizontal rules are visible only if HRuleWidth [916]>0

Note about rules.

Default value:

rvclWindowText [1105]

See also:
- Undo of assignments to table properties;
- Table rules [200] (with pictures).

See also properties:
- VRuleColor [922];
- HRuleWidth [916];
- HOutermostRule [915].

7.1.1.29 TRVTableItemInfo.HRuleWidth

Specifies the width of horizontal rules (horizontal lines between cells)

```
property HRuleWidth: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] 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[200] (with pictures).

See also properties

- VRuleWidth[922];
- HRuleColor[915];
- HOutermostRule[915].

7.1.1.30 TRVTableItemInfo.Opacity

Opacity of table background.

```
property Opacity: TRVOpacity[1082];
```

(introduced in version 16)

This value is used if Color[914]<>*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[199].

7.1.1.31 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 469 or TDBRichViewEdit 616)

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 469 or TDBRichViewEdit 616)
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 247.
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 955 properties of cells.

Option	Meaning
	But (unlike *rvtoIgnoreContentWidth*), heights of cells having BestHeight=0 are still calculated basing on their content. If you want to turn this option on only for some cells, use the cell's IgnoreContentHeight[958] property instead.

Visual Options

Option	Meaning
rvtoHideGridLines	If not set (default), invisible borders are shown using GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle[750] properties of the linked TRVStyle. It's not recommended to use this option. Use rvoShowGridLines option in TRichView.Options[247] instead.
rvtoCellBelowBorders	If not set (default), space below invisible side[963] of border has color[914] of tables. If set, color of cell[957] is used. Standard mode "Cells below borders"
rvtoOverlappingCorners	If not set (default), colored borders (CellBorderStyle[910] =*rvtbColor*) have three-dimensional appearance. If not set, they are flat. The effect can be noticed for wide borders with some invisible sides[963]. Standard corners Overlapping corners

RTF and DocX Export Options

Option	Meaning
rvtoRTFAllowAutofit	If set, tables with default width (BestWidth[906]=0) are exported to RTF and DocX as fit-by-content tables. Off by default. See also: RichViewTableDefaultRTFAutofit[1117]
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[209].

7.1.1.32 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[908] and CellBorderStyle[910]) 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[882]
rvtoContinue	If set, and the table has heading rows[914], heading rows are displayed only from the second table page

Default value:

[*rvtoHalftoneBorders,rvtoRowsSplit*]

See also:

- HeadingRowCount[914].

7.1.1.33 TRVTableItemInfo.RowCount

Returns the number of rows in the table.

```
property RowCount: Integer
```

(introduced in v10)

See also properties:

- ColCount[914];
- Cells[912];
- Rows[920].

7.1.1.34 TRVTableItemInfo.Rows

Provides indexed access to all rows in the table.

```
property Rows: TRVTableRows[968];
```

Rows is a list of table rows (TRVTableRow[966]), each row is a list of cells (TRVTableCellData[951]). Each row has the same number of cells.

The specific cell can be accessed as Rows[<Row>][<Col>] as well as Cells[912][<Row>,<Col>].

Use Rows to access properties of rows, for example Rows[<Row>].VAlign[968].Otherwise, Cells[912] is more convenient.

See also:

- Table Cells Overview[198]

7.1.1.35 TRVTableItemInfo.TextColSeparator

Value of this property is used when saving table in text file. This string will be written between cells.

```
property TextColSeparator: TRVUnicodeString[1099];
```

(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[921].

See also methods of TRichView:

- SaveText[352];
- GetSelText[321].

See also:

- Export of Tables [209].

7.1.1.36 TRVTableItemInfo.TextRowSeparator

Value of this property is used when saving table in text file. This string will be written between rows.

```
property TextRowSeparator: TRVUnicodeString[1099];
```

(changed in version 18)

Default value:

#13#10 (CR+LF)

See also properties:

- TextColSeparator [920].

See also methods of TRichView:

- SaveText [352];
- GetSelText [321].

See also:

- Export of Tables [209].

7.1.1.37 TRVTableItemInfo.VisibleBorders

Allows to hide some sides of table border.

```
property VisibleBorders: TRVBooleanRect[1018];
```

(introduced in v10)

Direct assignment to this property cannot be undone/redone by user (use SetTableVisibleBorders [946])

Default value:

True, *True*, *True*, *True* (all sides are visible)

See also properties:

- BorderWidth [909];
- BorderColor [906];
- BorderLightColor [907];
- BorderStyle [908].

See also properties of table cell:

- VisibleBorders [963].

7.1.1.38 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 [922]>0)

Note about rules.

Default value:

False

See also:

- Undo of assignments to table properties;
- Table rules [200] (with pictures).

See also properties:

- HOutermostRule [915];
- VRuleWidth [922];
- VRuleColor [922].

7.1.1.39 TRVTableItemInfo.VRuleColor

Specifies the color of vertical rules (vertical lines between cells)

```
property VRuleColor: TRVColor[1063];
```

Vertical rules are visible only if VRuleWidth [922]>0

Note about rules.

Default value:

rvclWindowText [1105]

See also:

- Undo of assignments to table properties;
- Table rules [200] (with pictures).

See also properties:

- HRuleColor [915];
- VRuleWidth [922];
- VOutermostRule [921].

7.1.1.40 TRVTableItemInfo.VRuleWidth

Specifies the width of vertical rules (vertical lines between cells)

```
property VRuleWidth: TRVStyleLength[1094];
```

This value is measured in Units [764] of the linked TRVStyle [739] 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 [200] (with pictures).

See also properties:

- HRuleWidth [916];
- VRuleColor [922];
- VOutermostRule [921].

7.1.1.41 TRVTableItemInfo's row and column colors

A set of properties defining default colors of cells in specific rows and columns.

```
property HeadingRowColor: TRVColor[1063];
property LastRowColor: TRVColor[1063];
property OddRowsColor: TRVColor[1063];
property EvenRowsColor: TRVColor[1063];
property FirstColumnColor: TRVColor[1063];
property LastColumnColor: TRVColor[1063];
property OddColumnsColor: TRVColor[1063];
property EvenColumnsColor: TRVColor[1063];
```

(introduced in version 17)

By default, all cells are transparent, and you can see the table background through cells (see Color [914] 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 [957] property.

Property	Meaning
HeadingRowColor	Color of heading rows [914]. 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 [913] columns.

Row bands are started from the first column after the table heading rows. If HeadingRowCount [914] >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[913] 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

rvclNone[1105]

See also

- Color[914]
- CellOverrideColor[912];
- HeadingRowCount[914]
- ColBandSize, RowBandSize[913]
- Cell.Color[957]
- Cell.Opacity[958]

7.1.2 Methods

In TRVTableItemInfo

AssignProperties[925]
CanMergeCells[925]
CanMergeSelectedCells[926]
Changed[927]
Create[927]
CreateEx[928]
DeleteCols[928]
DeleteEmptyCols[929]
DeleteEmptyRows[929]
DeleteRows[929]
DeleteSelectedRows[930]
DeleteSelectedCols[930]
Deselect[931]
Destroy[931]
EditCell[931]
GetCellAt[932]
GetEditedCell[932]
GetNormalizedSelectionBounds[933]
GetSelectionBounds[933]
InsertCols[934]
InsertColsLeft[934]
InsertColsRight[935]
InsertRows[935]
InsertRowsAbove[936]
InsertRowsBelow[936]
IsCellSelected[936]
LoadFromStream[937]

MergeCells 937
MergeSelectedCells 938
MoveRows 938
SaveRowsToStream 939
SaveToStream 939
Select 939
SelectCols 940
SelectRows 940
SetBackgroundProperty 940
SetCellBackgroundImage 941
SetCellBackgroundImageFileName 941
SetCellBackgroundProperty 941
SetCellBestHeight 942
SetCellBestWidth 942
SetCellBorderColor 942
SetCellBorderLightColor 943
SetCellColor 943
SetCellHint 943
SetCellOpacity 943
SetCellOptions 944
SetCellRotation 944
SetCellTag 944
SetCellVAlign 945
SetCellVisibleBorders 945
SetRowKeepTogether 945
SetRowPageBreakBefore 945
SetRowVAlign 945
SetTableVisibleBorders 946
SplitSelectedCellsHorizontally 946
SplitSelectedCellsVertically 946
UnmergeCells 947
UnmergeSelectedCells 947

7.1.2.1 TRVTableItemInfo.AssignProperties

Assigns all table properties from the **Source**.

```
procedure AssignProperties(Source: TRVTableItemInfo 901);
```

(introduced in v1.9)

Cells and their contents are not assigned.

7.1.2.2 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[912][**TopRow**, **TopCol**] with ColSpan[957] =**ColSpan** and RowSpan[962]=**RowSpan**.

Merging can be done, if

1. Cells[912][**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[912][**TopRow**, **TopCol**].ColSpan[957] < **ColSpan** or Cells[912][**TopRow**, **TopCol**].RowSpan[962] < **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[929] and DeleteEmptyCols[929] .

See also:

- CanMergeSelectedCells[926];
- MergeCells[937];
- MergeSelectedCells[938].

See also properties:

- ColCount[914];
- RowCount[920].

7.1.2.3 TRVTableItemInfo.CanMergeSelectedCells

Returns: is cell-merging of the selected cells possible?

```
function CanMergeSelectedCells(AllowMergeRC: Boolean): Boolean;
```

The same as CanMergeCells[925], 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[929] and DeleteEmptyCols[929].

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[925];
- MergeSelectedCells[938];
- MergeCells[937].

7.1.2.4 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[503]/EndItemModify[511] (if the table was inserted in TRichViewEdit[469]) or Format[295] (if the table was inserted in TRichView[216]).

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[304](ItemNo));
table.Cells[0,0].Clear[286];
table.Cells[0,0].AddNL[965]('One more line',0,0);
table.Changed;
MyRichView.Format[295];
```

Note: since version 1.7, it's not required to call table.Changed before Format[295]. But it is still useful when using BeginItemModify[503]/EndItemModify[511].

7.1.2.5 TRVTableItemInfo.Create

Creates a new TRVTableItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], 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[928].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.1.2.6 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[254] property of TRichView[216] or TRichViewEdit[469], 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[920]>0.

Example

```
var table:TRVTableItemInfo;
...
table := TRVTableItemInfo.CreateEx(5, 5, MyRichViewEdit.RVData[254]);
table.Color[914] := clYellow;
table.Cells[912][0,0].AddNL[965]('Hello',0,0);
MyRichViewEdit.InsertItem[531]('5x5 table', table);
```

See also methods:

- Create[927].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.1.2.7 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[914]-1.

Count – number of columns to delete.

If **DecreaseWidth** = *True*, BestWidth[955]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:

- Table operations[205];

- Undo note.

See also methods:

- DeleteSelectedCols [930];
- DeleteRows [929];
- DeleteEmptyCols [929].

7.1.2.8 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 [928] with *DecreaseWidth* = *False*.

The method keeps selection in the table.

See also:

- Table operations [205];
- Undo note.

See also methods:

- DeleteEmptyRows [929];
- DeleteSelectedCols [930];
- DeleteCols [928].

7.1.2.9 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 [929] with DecreaseHeight = *False*.

The method keeps selection in the table.

See also:

- Table operations [205];
- Undo note.

See also:

- DeleteEmptyCols [929];
- DeleteSelectedRows [930];
- DeleteRows [929].

7.1.2.10 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[920]-1.

Count – number of rows to delete.

if **DecreaseHeight** = *True*, BestHeight[955]s 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:

- Table operations[205];
- Undo note.

See also methods:

- DeleteSelectedRows[930];
- DeleteCols[928];
- DeleteEmptyRows[929].

7.1.2.11 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.

This method does nothing if selected columns contain items protected from deletion.

See also:

- Table operations[205];
- Undo note.

See also methods:

- DeleteSelectedRows[930];
- DeleteCols[928];
- DeleteEmptyCols[929].

7.1.2.12 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.

This method does nothing if selected rows contain items protected from deletion.

See also:

- Table operations [205];
- Undo note.

See also methods:

- DeleteSelectedCols [930];
- DeleteCols [928];
- DeleteEmptyCols [929].

7.1.2.13 TRVTableItemInfo.Deselect

Removes selection from the table.

```
procedure Deselect;
```

All selected cells become not selected.

See also:

- Overview of selection in tables [201].

See also methods:

- Select [939];
- SelectRows [940];
- SelectCols [940].

7.1.2.14 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.

7.1.2.15 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 [920]-1, **Col** must be in range 0..ColCount [914]-1.

Cells [931][**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 [931][Row, Col].Edit [1016].

This method must be called when the document is formatted.

See also:

- Overview of cell editing [203].

See also methods:

- GetNormalizedSelectionBounds [933];
- GetEditedCell [932].

7.1.2.16 TRVTableItemInfo.GetCellAt

Returns cell at the specified coordinates.

```
function GetCellAt(X,Y: TRVCoord[1064]; out Row,Col: Integer;
  IgnoreBorders: Boolean = False): Boolean;
```

Input parameters:

X, **Y** – coordinates, relative to the top left corner of the table.

If **IgnoreBorders** = *False*, the coordinates must be strictly inside the cell. Otherwise, (**X**, **Y**) may be inside a cell border or close to it.

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 [306].

7.1.2.17 TRVTableItemInfo.GetEditedCell

Returns inplace editor for the currently edited cell, or *nil* if there is no such cell.

```
function GetEditedCell(out Row,Col: Integer): TCustomRichViewEdit[469];
```

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 [201];
- Overview of cell editing [203].

See also methods:

- GetNormalizedSelectionBounds [933];
- EditCell [931].

7.1.2.18 TRVTableItemInfo.GetNormalizedSelectionBounds

Returns selection in table in convenient form.

```
function GetNormalizedSelectionBounds(IncludeEditedCell: Boolean;
         out 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[912][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[936].

Some operations are possible only on rectangular selection (for example, cell merging). It can be checked using CanMergeSelectedCells[926].

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[201]

See also methods:

- GetSelectionBounds[933];
- GetEditedCell[932].

7.1.2.19 TRVTableItemInfo.GetSelectionBounds

Returns selection in table.

```
function GetSelectionBounds(out StartRow, StartCol,
  RowOffs, ColOffs: Integer): Boolean;
```

Output parameters:

Cells[912][**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 926.

This method returns selection as it is stored internally. For practical use, GetNormalizedSelectionBounds 933 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 201.

See also methods:

- GetNormalizedSelectionBounds 933;
- Select 939;
- Deselect 931.

7.1.2.20 TRVTableItemInfo.InsertCols

Inserts **Count** columns at the position specified by **Index**.

```
procedure InsertCols(Index, Count, CopyIndex: Integer);
```

Parameters:

Index – index of the column to insert before (or ColCount 914 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:

- Table operations 205;
- Undo note.

See also methods:

- InsertRows 935;
- InsertColsLeft 934, InsertColsRight 935;
- DeleteCols 928.

7.1.2.21 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 an edited cell.

This method must be called when the document is formatted.

See also:

- Table operations [205];
- Undo note.

See also methods:

- InsertColsRight [935];
- InsertCols [934];
- InsertRowsAbove [936], InsertRowsBelow [936];
- DeleteSelectedCols [930].

7.1.2.22 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 an edited cell.

This method must be called when the document is formatted.

See also:

- Table operations [205];
- Undo note.

See also methods:

- InsertColsLeft [934];
- InsertCols [934];
- InsertRowsAbove [936], InsertRowsBelow [936];
- DeleteSelectedCols [930].

7.1.2.23 TRVTableItemInfo.InsertRows

Inserts **Count** rows at the position specified by **Index**.

```
procedure InsertRows(Index, Count, CopyIndex: Integer);
```

Parameters:

Index – index of the row to insert before (or RowCount [920] 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:

- Table operations [205];
- Undo note.

See also methods:

- InsertCols [934];
- InsertRowsAbove [936], InsertRowsBelow [936];
- DeleteRows [929].

7.1.2.24 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 an edited cell.

This method must be called when the document is formatted.

See also:

- Table operations [205];
- Undo note.

See also methods:

- InsertRowsBelow [936];
- InsertRows [935];
- InsertColsLeft [934], InsertColsRight [935];
- DeleteSelectedRows [930].

7.1.2.25 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 an edited cell.

This method must be called when the document is formatted.

See also:

- Table operations [205];
- Undo note.

See also methods:

- InsertRowsAbove [936];
- InsertRows [935];
- InsertColsLeft [934], InsertColsRight [935];
- DeleteSelectedRows [930].

7.1.2.26 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[920]-1 do
  for c := 0 to table.ColCount[914]-1 do
    if (table.Cells[912][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 [201].

See also methods:

- GetNormalizedSelectionBounds [933].

7.1.2.27 TRVTableItemInfo.LoadFromStream

Loads table from **Stream**.

```
procedure LoadFromStream(Stream: TStream);
```

Table must be saved using SaveToStream [939] or SaveRowsToStream [939] methods. Table can be loaded correctly only if collections of text, paragraph and list styles in TRVStyle [739] 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 [129] 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[531]('', Table) then
  Table.AfterLoading(rvlfRVF);
```

7.1.2.28 TRVTableItemInfo.MergeCells

Merges the specified cells in one.

```
procedure MergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer; AllowMergeRC:
```

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 [912][**TopRow**, **TopCol**] having ColSpan [957]=**ColSpan** and RowSpan [962] =**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 [925].

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 [929] and DeleteEmptyCols [929] (the latter is recommended).

See also:

- Table operations [205];
- Undo note.

See also methods:

- CanMergeCells [925];
- MergeSelectedCells [938];
- GetNormalizedSelectionBounds [933];
- UnmergeCells [947].

7.1.2.29 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 [926].

AllowMergeRC = *False* forbids to perform merging, if it will cause appearing rows or columns containing only *nil*s. It's not recommended to have such rows/columns, so you can either forbid such operation or delete them with DeleteEmptyRows [929] and DeleteEmptyCols [929] (the latter is recommended).

This method must be called when the document is formatted.

See also:

- Table operations [205];
- Undo note.

See also methods:

- CanMergeSelectedCells [926];
- MergeCells [937];
- GetNormalizedSelectionBounds [933];
- UnmergeSelectedCells [947].

7.1.2.30 TRVTableItemInfo.MoveRows

Moves **ARowCount** rows, starting from **AFromRow** row, to the new position (specified in **AToRow**).

```
function MoveRows(AFromRow, ARowCount, AToRow: Integer): Boolean;
```

(introduced in version 20)

Parameters:

AFromRow – the first row to move (in range 0 .. RowCount [920] - **ARowCount**)

ARowCount specifies how many rows will be moved (a positive value)

AToRow – where to move (in range 0 .. RowCount [920] - 1)

If **AToRow** < **AFromRow**, the moved rows are inserted before the current **AToRow**-th row.

If **AToRow** > **AFromRow**, the moved rows are inserted after the current **AToRow**-th row.

If values of the parameters are not in valid ranges, the method raises an exception.

Cells in the moved rows must not be vertically merged with upper and lower rows. Cells must not be merged vertically across the place of insertion.

If the method cannot move the rows because of cell merging, or if **ARowTo** defines the position inside the moved rows, the function returns *False*.

See also:

- Table operations [205];
- Undo note.

7.1.2.31 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 [939].

The result will be correct even if there are cells that overlap these rows because of RowSpan [962]>1.

See also method:

- LoadFromStream [937].

See also:

- Example how to divide table in two parts [214].

7.1.2.32 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 [129] with main documents.

See also method:

- LoadFromStream [937];
- SaveRowsToStream [939].

7.1.2.33 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 [933].

Selection includes all cells intersected with imaginary rectangle that fully includes Cells [912] [**StartRow**, **StartCol**] and Cells [912] [**StartRow**+**RowOffs**, **StartCol**+**ColOffs**].

You cannot remove selection with this method, because Select(StartRow, StartCol, 0,0) selects one cell, use Deselect [931].

This method must be called when the document is formatted.

See also:

- Overview of selection in tables [201]

See also methods:

- SelectRows [940], SelectCols [940];
- GetSelectionBounds [933];
- Deselect [931].

7.1.2.34 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[914]-1.

Count – number of columns to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables[201]

See also methods:

- SelectRows[940],
- Select[939].

7.1.2.35 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[920]-1.

Count – number of rows to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables[201]

See also methods:

- SelectCols[940];
- Select[939].

7.1.2.36 TRVTableItemInfo.SetBackgroundProperty

Assigns **Value** to the property of BackgroundProperties[905] identified by **PropId**, as an editing operation.

```
procedure SetBackgroundProperty(
  PropId: TRVBackgroundPropertyId[1056];
  Value: NativeInt);
```

Any assignment to properties of BackgroundProperties[905] can be undone/redone, if it was called for a table inserted in TRichViewEdit (after insertion[531]).

This method was added to allow working with table properties in the same way as with cell properties (see SetCellBackgroundProperty[941]).

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.37 TRVTableItemInfo.SetCellBackgroundImage

Assigns **Value** to Cells [912][**Row**, **Col**].BackgroundImage [953], as an editing operation.

```
procedure SetCellBackgroundImage(Value: TRVGraphic[1032]; Row,Col: Integer);
```

Unlike a direct assignment to Cells [912][**Row**, **Col**].BackgroundImage [953] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion [531]).

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 [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.38 TRVTableItemInfo.SetCellBackgroundImageFileName

Assigns **Value** to Cells [912][**Row**, **Col**].BackgroundImageFileName [953], as an editing operation.

```
procedure SetCellBackgroundImageFileName(
  const Value: TRVUnicodeString[1099]; Row,Col: Integer);
```

(changed in version 18)

Unlike a direct assignment to Cells [912][**Row**, **Col**].BackgroundImageFileName [953] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.39 TRVTableItemInfo.SetCellBackgroundProperty

Assigns **Value** to the property of Cells [912][**Row**, **Col**].BackgroundProperties [954] identified by **PropId**, as an editing operation.

```
procedure SetCellBackgroundProperty(
  PropId: TRVBackgroundPropertyId[1056];
  Value: NativeInt; Row, Col: Integer);
```

Unlike a direct assignment to properties of Cells [912][**Row**, **Col**].BackgroundProperties [954], this method can be undone/redone, if it was called for a table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- SetBackgroundProperty [940]
- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.40 TRVTableItemInfo.SetCellBackgroundStyle

Assigns **Value** to Cells[912][**Row**, **Col**].BackgroundStyle[954], as an editing operation.

```
procedure SetCellBackgroundStyle(Value: TRVItemBackgroundStyle[1080];
  Row,Col: Integer);
```

This method is deprecated, use SetCellBackgroundProperty[941] instead.

Unlike a direct assignment to Cells[912][**Row**, **Col**].BackgroundStyle[954] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208]

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.41 TRVTableItemInfo.SetCellBestHeight

Assigns **Value** to Cells[912][**Row**, **Col**].BestHeight[955], as an editing operation.

```
procedure SetCellBestHeight(Value: TRVStyleLength[1094]; Row,Col: Integer);
```

Unlike a direct assignment to Cells[912][**Row**, **Col**].BestHeight[955] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208]

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.42 TRVTableItemInfo.SetCellBestWidth

Assigns **Value** to Cells[912][**Row**, **Col**].BestWidth[955], as an editing operation.

```
procedure SetCellBestWidth(Value: TRVHTMLLength[1079]; Row,Col: Integer);
```

Unlike a direct assignment to Cells[912][**Row**, **Col**].BestWidth[955] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.43 TRVTableItemInfo.SetCellBorderColor

Assigns **Value** to Cells[912][**Row**, **Col**].BorderColor[956], as an editing operation.

```
procedure SetCellBorderColor(Value: TRVColor[1063]; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells[912][**Row**, **Col**].BorderColor[956] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.44 TRVTableItemInfo.SetCellBorderLightColor

Assigns **Value** to Cells [912][**Row**, **Col**].BorderLightColor [956], as an editing operation.

```
procedure SetCellBorderLightColor(Value: TRVColor[1063]; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells [912][**Row**, **Col**].BorderLightColor [956] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.45 TRVTableItemInfo.SetCellColor

Assigns **Value** to Cells [912][**Row**, **Col**].Color [957], as an editing operation.

```
procedure SetCellColor(Value: TRVColor[1063]; Row,Col: Integer);
```

Unlike a direct assignment to Cells [912][**Row**, **Col**].Color [957] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.46 TRVTableItemInfo.SetCellHint

Assigns **Value** to Cells [912][**Row**, **Col**].Hint [958], as an editing operation.

```
procedure SetCellHint(const Value: TRVUnicodeString[1099]; Row,Col: Integer);
```

Unlike a direct assignment to Cells [912][**Row**, **Col**].Hint [958] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.47 TRVTableItemInfo.SetCellOpacity

Assigns **Value** to Cells [912][**Row**, **Col**].Opacity [958], as an editing operation.

```
procedure SetCellOpacity(Value: TRVOpacity[1082]; Row,Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells[912][**Row**, **Col**].Opacity[958] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.48 TRVTableItemInfo.SetCellOptions

Assigns **Value** to Cells[912][**Row**, **Col**].Options[959], as an editing operation.

```
procedure SetCellOpacity(Value: TRVTableCellOptions[959]; Row,Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells[912][**Row**, **Col**].Options[959] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.49 TRVTableItemInfo.SetCellRotation

Assigns **Value** to Cells[912][**Row**, **Col**].Rotation[961], as an editing operation.

```
procedure SetCellRotation(Value: TRVDocRotation[1065]; Row,Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells[912][**Row**, **Col**].Rotation[961] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120].

7.1.2.50 TRVTableItemInfo.SetCellTag

Assigns **Value** to Cells[912][**Row**, **Col**].Tag[963], as an editing operation.

```
procedure SetCellVAlign(Value: Tag[963]; Row,Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells[912][**Row**, **Col**].Tag[963] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[531]).

Use this method in conjunction with methods for updating editor window, see example[208].

See also:

- Undo/redo in tables[208];
- Undo/redo in TRichViewEdit[120]

- Tags [96]

7.1.2.51 TRVTableItemInfo.SetCellVAlign

Assigns **Value** to Cells [912][**Row**, **Col**].VAlign [963], as an editing operation.

```
procedure SetCellVAlign(Value: TRVCellVAlign[1061]; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells [912][**Row**, **Col**].VAlign [963] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.52 TRVTableItemInfo.SetCellVisibleBorders

Assigns values to Cells [912][**Row**, **Col**].VisibleBorders [963], as an editing operation.

```
procedure SetCellVisibleBorders(Left, Top, Right, Bottom: Boolean;
  Row,Col: Integer);
```

(introduced in version 1.6)

Unlike direct assignment to Cells [912][**Row**, **Col**].VisibleBorders [963] subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.53 TRVTableItemInfo.SetRow*

Assigns **Value** to Rows [920][**Row**].VAlign [968] (or PageBreakBefore [967], or KeepTogether [967]), as an editing operation.

```
procedure SetRowVAlign(Value: TRVCellVAlign[1061]; 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 [920][**Row**] property, these methods can be undone/redone, if they were called for a table inserted in TRichViewEdit (after insertion [531]).

Use these methods in conjunction with the methods for updating an editor window, see the example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.54 TRVTableItemInfo.SetTableVisibleBorders

Assigns values to VisibleBorders [921], as an editing operation.

```
procedure SetTableVisibleBorders(Left, Top, Right, Bottom: Boolean);
```

(introduced in version 10)

Unlike direct assignment to VisibleBorders [921] subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion [531]).

Use this method in conjunction with methods for updating editor window, see example [208].

See also:

- Undo/redo in tables [208];
- Undo/redo in TRichViewEdit [120].

7.1.2.55 TRVTableItemInfo.SplitSelectedCellsHorizontally

Splits each selected cell into **RowCount** cells (inserting horizontal splitters).

```
procedure SplitSelectedCellsHorizontally(RowCount: Integer);
```

This method splits cells calling UnmergeCells [947], InsertRows [935] and MergeCells [937]. This method can change number of rows.

It's recommended to call this method only when selection has a rectangular shape, see CanMergeSelectedCells [926].

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:

- Table operations [205];
- Undo note.

See also methods:

- SplitSelectedCellsVertically [946];
- UnmergeSelectedCells [947];
- GetNormalizedSelectionBounds [933].

7.1.2.56 TRVTableItemInfo.SplitSelectedCellsVertically

Splits each selected cell into **ColCount** cells (inserting vertical splitters).

```
procedure SplitSelectedCellsVertically(ColCount: Integer);
```

This method splits cells calling UnmergeCells [947], InsertCols [934] and MergeCells [937]. This method can change number of columns.

It's recommended to call this method only when selection has a rectangular shape, see CanMergeSelectedCells [926].

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:

- Table operations [205];

- Undo note.

See also methods:

- SplitSelectedCellsVertically 946;
- UnmergeSelectedCells 947;
- GetNormalizedSelectionBounds 933.

7.1.2.57 TRVTableItemInfo.UnmergeCells

Unmerges cells.

```
procedure UnmergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer;
  UnmergeRows, UnmergeCols: Boolean);
```

Cells can be merged using the methods MergeCells 937 or MergeSelectedCells 938.

This method unmerges all cells in range from Cells 912 [**TopRow**, **LeftCol**] to Cells 912 [**TopRow**+**RowSpan**, **LeftCol**+**ColSpan**].

If **UnmergeRows**=*True*, it unmerges each cell into RowSpan 962 rows.

If **UnmergeCols**=*True*, it unmerges each cells into ColSpan 957 columns.

UnmergeRows and *UnmergeCells* can both be *True*.

See also:

- Table operations 205;
- Undo note.

See also methods:

- UnmergeSelectedCells 947;
- MergeCells 937.

7.1.2.58 TRVTableItemInfo.UnmergeSelectedCells

Unmerges the selected cells.

```
procedure UnmergeSelectedCells(UnmergeRows, UnmergeCols: Boolean);
```

Cells can be merged using the methods MergeCells 937 or MergeSelectedCells 938.

If **UnmergeRows**=*True*, it unmerges each selected cell into RowSpan 962 rows.

If **UnmergeCols**=*True*, it unmerges each selected cells into ColSpan 957 columns.

UnmergeRows and **UnmergeCells** can both be *True*.

This method must be called when the document is formatted.

See also:

- Table operations 205;
- Undo note.

See also methods:

- UnmergeCells 947;
- MergeSelectedCells 938;
- GetNormalizedSelectionBounds 933.

7.1.3 Events

Since tables are not components, events cannot be assigned at design time in object inspector.

They must be assigned from code, when a table is created.

In TRVTableItemInfo

OnCellEditing 948
OnDrawBackground 949
OnDrawBorder 949

7.1.3.1 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 912[**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 129 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 276 or InsertItem 531 methods, or is loaded from a file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo 899; var Text: String;
```

```
  RVData: TCustomRVData);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
    TRVTableItemInfo[901](Item).OnCellEditing := DoCellEditing;
end;
```

See also:

- TParaInfo.Options[833] (*rvpaoReadOnly, rvpaoStyleProtect, rvpaoDoNotWantReturns* – protection options for paragraphs);
- TFontInfo.Protection[815] (protection options for text);
- TRVExtraItemProperty[1067] (*rvepDeleteProtect, rvepResizable* – protection option for nontext items).

7.1.3.2 TRVTableItemInfo.OnDrawBackground

Allows custom drawing of table backgrounds

```
type
  TRVTableDrawBackEvent = procedure(Sender: TRVTableItemInfo[901];
    Canvas: TCanvas; Left, Top, Right, Bottom: TRVCoord[1064];
    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[949]

7.1.3.3 TRVTableItemInfo.OnDrawBorder

Allows custom drawing of table borders

```
type
  TRVTableDrawBorderEvent = procedure (Sender: TRVTableItemInfo;
    Canvas: TCanvas; Left,Top,Right,Bottom: TRVCoord[1064];
```

```
    Width: TRVStyleLength[1094];
    LightColor, Color, BackgroundColor: TRVColor[1063];
    Style: TRVTableBorderStyle[1096]; Printing: Boolean;
    VisibleBorders: TRVBooleanRect[1018]; 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[260].Units[764].

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[921] property for the table, or VisibleBorders[963] 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[910]=1, CellVSpacing[913]=0, CellHSpacing[911]=0.
  This example ignores VisibleBorders.
}
procedure TMyForm.DoDrawBorder(Sender: TRVTableItemInfo;
  Canvas: TCanvas; Left, Top, Right, Bottom: TRVCoord[1064];
  Width: TRVStyleLength[1094];
  LightColor, Color, BackgroundColor: TRVColor[1063];
  Style: TRVTableBorderStyle[1096];
  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[129] files, so you need to reassign them when tables are loaded. The best place to assign this event is TRichView.OnItemAction event.

```
// MyRichViewEdit.OnItemAction[387]. Assigns DoDrawBorder to
// OnDrawBorder events of all tables inserted in MyRichViewEdit.
// (this assignment occurs when table is inserted using
// AddItem[276] or InsertItem[531] methods, or is loaded from RVF or RTF file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView[216];
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo[899];
  var Text: TRVUnicodeString[1099]; RVData: TCustomRVData[1014]);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
    TRVTableItemInfo[901](Item).OnDrawBorder := DoDrawBorder;
end;
```

See also

- OnDrawBackground[949]

7.1.4 Classes of properties

Classes of TRVTableItemInfo[901] Properties

- TRVTableRows[968] – a list of table rows (TRVTableRow[966]). This is a type of Rows[920] property.
- TRVTableRow[966] – a list of table cells (TRVTableCellData[951]). Accessible as Rows[r][969].
- TRVTableCellData[951] – a table cell. Accessible as Cells[r, c][912]

7.1.4.1 TRVTableCellData

TRVTableCellData represents one cell in a table (see TRVTableItemInfo[901]). Cells can be accessed using Cells[912] property of table.

Unit RVTable.

Syntax

```
TRVTableCellData = class(TRVItemFormattedData[1014])
```

Hierarchy

TObject
TPersistent
TCustomRVData[1014]
TCustomRVFormattedData
TRVItemFormattedData

Methods and Properties

TRVTableCellData has many properties and methods corresponding to properties and methods [965] of TRichView [216].

TRVTableCellData and class of object TRichView.RVData [254] are derived from the same base class.

In addition, this class has properties affecting the cell view and layout:

- Color [957] – background color of cell;
- BackgroundImage [953], BackgroundProperties [954] – background image, its visibility, size, position, and repetition;
- BackgroundImageFileName [953] – can be used when exporting to HTML;
- BorderColor [956], BorderLightColor [956] – colors of border;
- BestWidth [955], BestHeight [955], IgnoreContentHeight [958] – preferred size of cell,
- VAlign [963] – vertical alignment of contents;
- Rotation [961] – rotation of contents;
- VisibleBorders [963] – defines visible sides of cell border.
- Tag [963] – a string value (not used by the component itself).

These properties can be saved with cell in RVF [129] file or stream.

Also, this class has read-only properties:

- ColSpan [957], RowSpan [962].

Important methods:

- GetRVData [964].

See also

Table Cells Overview [198]

7.1.4.1.1 Properties

Inherited from TRVItemFormattedData [1014]

Read here [965]

In TRVTableCellData

- BackgroundImage [953]
- BackgroundImageFileName [953]
- BackgroundProperties [954]
- BestHeight [955]
- BestWidth [955]
- BorderColor [956]
- BorderLightColor [956]
- Color [957]
- ColSpan [957]
- Hint [958]
- IgnoreContentHeight [958]
- Opacity [958]
- Rotation [961]

- RowSpan 962
- Tag 963
- VAlign 963
- VisibleBorders 963

7.1.4.1.1.1 TRVTableCellData.BackgroundImage

Background image for the table cell

```
property BackgroundImage: TRVGraphic 1032;
```

(introduced in v1.8)

Assignment to this property copies the image, so you still need to free the source image yourself.

Visibility, size, position and repetition of the image is defined in the BackgroundProperties 954 property.Transparent images are supported.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount 1114).

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 953;
- BackgroundProperties 954;
- Color 957.

See also properties of table:

- BackgroundImage 904;
- BackgroundProperties 905.

See also properties of TRichView:

- BackgroundPicture 229;
- BackgroundProperties 229.

7.1.4.1.1.2 TRVTableCellData.BackgroundImageFileName

This is a string for storing file name associated with BackgroundImage 953

```
property BackgroundImageFileName: TRVUnicodeString 1099;
```

(introduced in v1.9; changed in version 18)

If *rvhtmlsioUseItemImageFileNames* in HTMLSaveProperties 244.ImageOptions 442, 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 1072.

See also properties of table:

- BackgroundImageFileName [905].

7.1.4.1.1.3 TRVTableCellData.BackgroundStyle

Defines the position of BackgroundImage [953].

```
property BackgroundStyle: TRVItemBackgroundStyle[1080];
```

(introduced in v1.8)

Deprecated, use BackgroundProperties [954] instead.

See also:

- BackgroundImage [953];
- Color [957].

See also properties of table:

- BackgroundProperties [905];
- BackgroundImage [904].

See also properties of TRichView:

- BackgroundProperties [229];
- BackgroundPicture [229].

7.1.4.1.1.4 TRVTableCellData.BackgroundProperties

Defines visibility, size, position, and repetition of BackgroundImage [953].

```
property BackgroundProperties: TRVBackgroundProperties[1021];
```

(introduced in v24)

This property can be *nil*. In that case, all of its sub-properties should be considered to have their default values.

A table can contain a very large number of cells, and most of them may not have a background image. To avoid unnecessary memory usage, memory for the **BackgroundProperties** property is allocated only when there is a non-empty BackgroundImage [953] and/or when it is necessary to assign a property value different from the default. Use the methods BackgroundPropertiesAllocate and BackgroundPropertiesFinalize [964] to access this property.

A direct assignment to sub-properties of this property cannot be undone/redone by user, use table.SetCellBackgroundProperty [941].

See also:

- BackgroundImage [953];
- Color [957].

See also properties of table:

- BackgroundProperties [905];
- BackgroundImage [904].

See also properties of TRichView:

- BackgroundProperties [229];
- BackgroundPicture [229].

7.1.4.1.1.5 TRVTableCellData.BestHeight

Height for the cell.

```
property BestHeight: TRVStyleLength[1094];
```

This value is measured in Units[764] of the linked TRVStyle[739] component.

This property corresponds to <td height> in HTML, but there is a difference. **BestHeight** does not include CellVPadding[913], while <td height> includes vertical cell padding.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBestHeight[942].

By default, this value defines the minimum cell width; cell can have greater height if it is necessary to show its content.

The meaning of this property is changed if you assign IgnoreContentHeight[958] = *True* for this cell, or if *rvtoIgnoreContentHeight* is included in table.Options[916]. In this case, if this property has a positive value, it is used as a cell height; height of content of this cell is ignored (however, cell height may be greater if other cells in this row have greater heights). If this property has a zero value, height of cell is defined by its content.

Default value:

0

See also properties:

- BestWidth[955];
- IgnoreContentHeight[958].

See also properties of table:

- Options[916] (*rvtoIgnoreContentHeight* option).

See also:

- Table Resizing[211].

7.1.4.1.1.6 TRVTableCellData.BestWidth

Preferred width for the cell

```
property BestWidth: TRVHTMLLength[1079];
```

This is a preferred width for the cell, measured either in Units[764] of the linked TRVStyle[739] component 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[942].

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[911]*2 + paragraph indents (FirstIndent[831], LeftIndent[832], RightIndent[835]) (if *rvtoIgnoreContentWidth* is not included in table.Options[916]).

The cell can be narrower than specified in this property, if the table is not wide enough.

Default value:

0

See also properties:

- BestHeight[955].

See also properties of table:

- BestWidth[906];
- Options[916] (*rvtoIgnoreContentWidth* option).

See also:

- Table Resizing[211].

7.1.4.1.1.7 TRVTableCellData.BorderColor

Color of border for this cell.

```
property BorderColor: TRVColor[1063]
```

(introduced in version 1.6)

If this value of this property is *rvclNone*[1105], the cell uses table.CellBorderColor[909].

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[942].

Default value:

rvclNone[1105]

See also properties:

- BorderLightColor[956];
- Color[957].

See also properties of table:

- CellBorderColor[909];
- BorderColor[906];
- CellBorderStyle[910].

7.1.4.1.1.8 TRVTableCellData.BorderLightColor

"Light" color of 3d border for this cell

```
property BorderLightColor: TRVColor[1063];
```

(introduced in version 1.6)

If this value of this property is *rvclNone*[1105], the cell uses table.CellBorderLightColor[910].

This value defines light color of 3d border.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBorderLightColor[943].

Default value:

rvclNone[1105]

See also properties:

- BorderColor[956];

- Color[957].

See also properties of table:

- CellBorderColor[909];
- BorderColor[906];
- CellBorderStyle[910].

7.1.4.1.1.9 TRVTableCellData.Color

Background color of the cell

```
property Color: TRVColor[1063];
```

Set to *rvclNone*[1105] 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[943].

If this property is equal to clNone, a color of a row/column[923] can be used.

Default value:

rvclNone[1105]

See also properties:

- BorderColor[956];
- BorderLightColor[956].

See also properties of table:

- Color[914];
- CellOverrideColor[912];
- CellBorderColor[909];
- CellBorderLightColor[910].

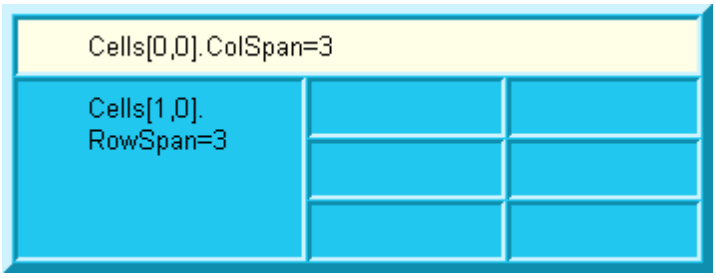
7.1.4.1.1.10 TRVTableCellData.ColSpan

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[962].

7.1.4.1.1.11 TRVTableCellData.Hint

Contains the text string that can appear when the user moves the mouse pointer over the cell.

```
property Hint: TRVUnicodeString[1099];
```

(introduced in version 10)

Default value:

'' (empty string)

See also:

- TRVExtraItemStrProperty[1072] (*rvespHint* defines items hints, including table hints);
- TCustomRichView.OnItemHint[388] (unlike item hints, cell hints cannot be altered in this event).

7.1.4.1.1.12 TRVTableCellData.IgnoreContentHeight

Defines the mode of cell height calculation.

```
property IgnoreContentHeight: Boolean;
```

(introduced in version 21)

If *False*, the height of cell is the maximum of BestHeight and a height of content in this cell.

If *True*:

- if BestHeight[955] > 0, it defines the cell height
- if BestHeight[955] = 0, content of this cell defines the cell height.

The actual cell height may be greater, because the a row has a height of its tallest cell.

This property is used if *rvtoIgnoreContentHeight* is not included in table.Options[916]. Otherwise, the table works as if all its cells have **IgnoreContentHeight** = *True*.

Default value:

False

See also:

- Table Resizing[211].

7.1.4.1.1.13 TRVTableCellData.Opacity

Opacity of cell background

```
property Opacity: TRVOpacity[1082];
```

(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[943].

The cell color is specified in Color[957] property. If Color[957]=*clNone*, the default row/column color[923] is used.

Default value:

100000 (i.e. 100%)

See also properties of table:

- Opacity[916]

7.1.4.1.1.14 TRVTableCellData.Options

Options for table cell

```
type
  TRVTableCellOption = (rvtcoReverseLineOrder, rvtcoUseVerticalFonts);
  TRVTableCellOptions = set of TRVTableCellOption;
property Options: TRVTableCellOptions;
```

(introduced in version 16)

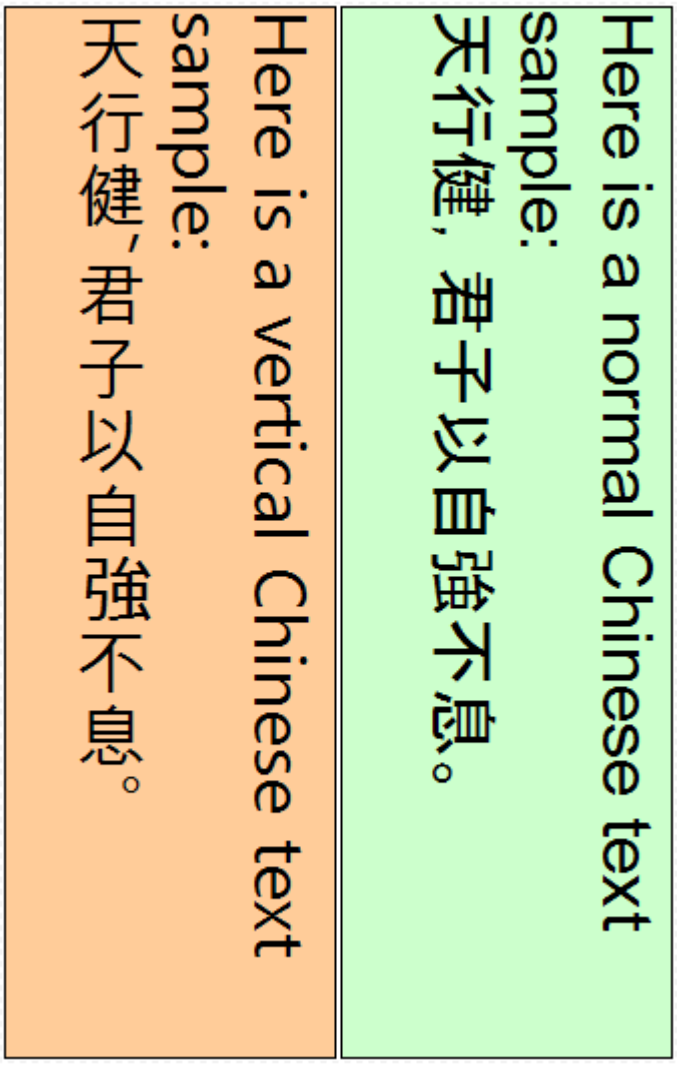
Option	Meaning
rvtcoReverseLineOrder	If set, lines in this cell are have reverse order. For example: • if Rotation[961]=*rvrotNone*, lines are ordered from bottom to top; • if Rotation[961]=*rvrot90*, lines are ordered from left to right
rvtcoUseVerticalFonts	If set, the component adds '@' to the beginning of font names to when applying fonts to canvas, if Rotation[961] 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[961]=*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[961]=*rvrot90*, VAlign[963]=*rvcBottom*, Options=[*rvtcoReverseLineOrder*]. In this script, columns are ordered from left to right.

Default value:

[]

7.1.4.1.1.15 TRVTableCellData.Rotation

Defines the rotation of the cell content.

```
property Rotation:TRVDocRotation[1065];
```

(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[944].

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 [959].

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 [174] 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 [129];
- 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 [963] property;
- Table.CellPadding [912]

Default value:

rvrotNone

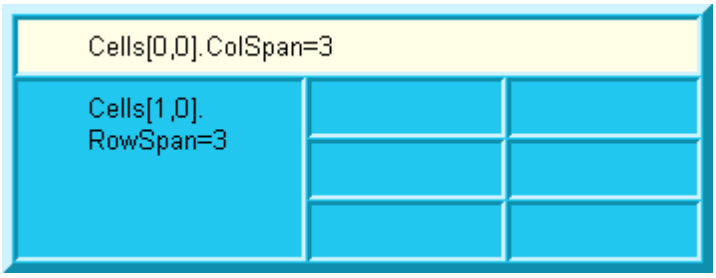
7.1.4.1.1.16 TRVTableCellData.RowSpan

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 [957].

7.1.4.1.1.17 TRVTableCellData.Tag

Contains additional string value associated with this table cell.

```
property Tag: TRVTag[1096];
```

(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[96]

7.1.4.1.1.18 TRVTableCellData.VAlign

Defines vertical alignment of the cell content.

```
property VAlign: TRVCellVAlign[1061];
```

(introduced in version 1.6)

If this property is set to *rvcVDefault*, the cell uses vertical alignment of its row, see TRVTableRow.VAlign[968].

A direct assignment to this property cannot be undone/redone by user, use table.SetCellVAlign[945].

If the cell is rotated[961], 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

7.1.4.1.1.19 TRVTableCellData.VisibleBorders

Allows to hide some sides of cell border.

```
property VisibleBorders: TRVBooleanRect[1018];
```

A direct assignment to this property cannot be undone/redone by user, use table.SetCellVisibleBorders[945].

Default value:

True, *True*, *True*, *True* (all sides are visible)

See also properties of table:

- VisibleBorders[921];
- CellBorderWidth[910];
- CellBorderColor[909];
- CellBorderLightColor[910].

7.1.4.1.2 Methods

Inherited from TRVItemFormattedData[1014]

Edit[1016]
GetOriginEx[1018]
GetSourceRVData[1017]
Also read here[965]

In TRVTableCellData

BackgroundPropertiesAllocate[964]
BackgroundPropertiesFinalize[964]
GetRVData[964]

7.1.4.1.2.1 TRVTableCellData.BackgroundPropertiesAllocate, BackgroundPropertiesFinalize

These methods allow working with the BackgroundProperties[954] property.

```
procedure BackgroundPropertiesAllocate;
procedure BackgroundPropertiesFinalize;
```

A table can contain a very large number of cells, and most of them may not have a background image. To avoid unnecessary memory usage, memory for the BackgroundProperties[954] property is allocated only when there is a non-empty BackgroundImage[953] and/or when it is necessary to assign a property value different from the default.

Working with the BackgroundProperties[954] property is done as follows.

First, call **BackgroundPropertiesAllocate**; if the BackgroundProperties[954] object has not been created yet, this method will create it.

Then you can access the BackgroundProperties[954] properties.

When finished, call **BackgroundPropertiesFinalize**; if there is no background image and all BackgroundProperties[954] values are equal to their defaults, this method will free the memory allocated for BackgroundProperties[954].

7.1.4.1.2.2 TRVTableCellData.GetRVData

Returns object containing cell data (document items)

```
function GetRVData: TCustomRVData[1014];
```

The cell is "an object containing data" itself. But when cell is in editing state[203], 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[254] property of inplace editor, otherwise.

So when working with content of cell directly[965], 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[1014];
begin
  Result := '';
  for r := 0 to table.RowCount[920]-1 do
  begin
    for c := 0 to table.ColCount[914]-1 do
      if table.Cells[912][r,c]<>nil then
      begin
        RVData := table.Cells[r,c].GetRVData;
        for i := 0 to RVData.ItemCount[244]-1 do
        begin
          if (i>0) and RVData.IsFromNewLine[325](i) then
            Result := Result + #13#10;
          if RVData.GetItemStyle[310](i)=rvsTab then
            Result := Result + #9
           else if RVData.GetItemStyle(i)>=0 then
            Result := Result + RVData.GetItemText[310](i);
        end;
        Result := Result + table.TextColSeparator[920];
      end;
    Result := Result + table.TextRowSeparator[921];
  end;
end;
```

See also methods:

- GetSourceRVData[1017].

7.1.4.1.3 Additional properties of TRVTableCellData

TRVTableCellData[951] has many properties and methods, similar to properties and methods of TRichView[216].

Some of them are listed below.

But when the cell is in editing state, RichViewEdit[469] 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[964] 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[196].

See also: TRVTableItemInfo.Changed[927]

Cell methods for appending items

(links are to the corresponding methods of TCustomRichView)

- AddBreak[269];
- AddBullet[270];
- AddControl[272];

- AddHotspot[275];
- AddFmt[273], AddItem[276];
- AddNL[277];
- AddPicture[278];

(note: it's not recommended to use the methods of the group above for cells of table inserted in TRichViewEdit[469], if editor's UndoLimit[490]<>0)

Obtaining information from cell

- GetItemStyle[310];
- GetBreakInfo[296], GetBulletInfo[297], GetControlInfo[301], GetHotspotInfo[303], GetPictureInfo[318], GetTextInfo[321];
- GetItemTag[310], IsParaStart[325], GetItemPara[309], IsFromNewLine[325], GetOffsBeforeItem[317], GetOffsAfterItem[316].

Modifying items

SetBreakInfo[358], SetBulletInfo[359], SetControlInfo[361], SetHotspotInfo[364], SetPictureInfo[370], SetItemTag[367].

(note: it's not recommended to use the methods of the group above for cells of table inserted in TRichViewEdit[469], if editor's UndoLimit[490]<>0)

7.1.4.2 TRVTableRow

This class represents one row in table (TRVTableItemInfo)[901]

Unit RVTable;

Syntax

```
TRVTableRow = class(TRVDataList)
```

Hierarchy

TObject
TList
TRVList
TRVDataList

Using

Table has Rows[920] property of class TRVTableRows[968]. 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[901] instead.

Properties

The key properties of this class are Items[967] and Count. Count is the number of columns in the table (all rows in the table have the same number of cells).

VAlign[968] defines the default vertical alignment for the table cells in this row.

PageBreakBefore[967] allows adding a page break before this row.

KeepWithNext[967] allows keeping the content of this row on a single page, if possible.

See Also

Table Cells Overview[198]

7.1.4.2.1 Properties

In TRVTableRow

- Items[967]
- KeepTogether[967]
- PageBreakBefore[967]
- VAlign[968]

Derived from TList

Count

7.1.4.2.1.1 TRVTableRow.Items

Provides indexed access to the cells of table

```
property Items[Index: Integer]:TRVTableCellData[951]; default;
```

Each item represents one cell of the table.

Usually cells are accessed as Cells[912] property of table instead.

7.1.4.2.1.2 TRVTableRow.KeepTogether

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[919].

The effect is similar to including *rvpaoKeepWithNext* in Options[833] 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[945].

Default value:

False

7.1.4.2.1.3 TRVTableRow.PageBreakBefore

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[919] is ignored.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowPageBreakBefore[945].

Default value:

False

7.1.4.2.1.4 TRVTableRow.VAlign

Defines default vertical alignment for contents of cells in the row

```
property VAlign:TRVCellVAlign[1061];
```

(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[945].

Default value:

rvcTop

See also properties of cell:

- VAlign[963]

7.1.4.2.2 Methods

(please do not call methods of TRVTableRow directly)

7.1.4.3 TRVTableRows

TRVTableRows represents contents of table (see TRVTableItemInfo[901])

Unit RVTable.

Syntax

```
TRVTableRows = class(TRVList)
```

Hierarchy

TObject
TList
TRVList

Using

This is the type of Rows[920] properties of table. TRVTableRows is a list of TRVTableRow[966]s which are in their order are lists of TRVTableCellData[951] (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[901] instead.

The key properties of this class are Items [969] and Count. Count is the number of rows in the table.

This class has method for finding main cell for given merged cell: GetMainCell [969].

See Also

Table Cells Overview [198]

7.1.4.3.1 Properties

In TRVTableRows

- Items [969]

Derived from TList

Count

7.1.4.3.1.1 TRVTableRows.Items

Provides indexed access to the rows of table

```
property Items[Index: Integer]: TRVTableRow[966]; default;
```

Each item represents one row of the table.

7.1.4.3.2 Methods

In TRVTableRows

GetMainCell [969]

(please do not call other methods of TRVTableRows directly)

7.1.4.3.2.1 TRVTableRows.GetMainCell

Returns the main (top left) cell for given merged cell.

```
function GetMainCell(ARow,ACol: Integer; out MRow, MCol: Integer): TRVTableCell
```

Some cells [912] in table [901] 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 [912][**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 [912][**MRow**, **MCol**]

7.2 Label - TRVLabelItemInfo

TRVLabelItemInfo is a class representing "label" [181] 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 [276]) or inserted into TRichViewEdit (see InsertItem [531]). Style [310] of this item type: *rvsLabel* (-200)

Unit RVLabelItem.

Syntax

```
TRVLabelItemInfo = class(TRVRectItemInfo)
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]
TRVNonTextItemInfo
TRVRectItemInfo

Properties

This item displays Text [972] using font defined in TextStyleNo [973]. ProtectTextStyleNo [972] can be used to protect TextStyleNo [973] from changing.

You can set minimal possible width for this item (MinWidth [971]) and horizontal text Alignment [971].

You can define Cursor [971] for this item.

Use SetCurrentItemExtraIntPropertyEx [566] and SetCurrentItemExtraStrPropertyEx [567] to change properties of label items as editing operations.

Related Properties of TRVStyle [739]

- FieldHighlightColor [746] – color for highlighting label items (overrides background color [808] specified in the text style)
- FieldHighlightType [747] specifies when to highlight label items.

Inherited Classes

The following item types are inherited from this class:

- TRVSeqItemInfo [978] (numbered sequences);
- TRVEndNoteItemInfo [985] (endnote);
- TRVFootnoteItemInfo [988] (footnote);
- TRVNoteReferenceItemInfo [1006] (reference to the parent footnote or endnote).

See Also...

Demos:

- Demos*\Assorted\Fields\MailMerge-LabelItems\

7.2.1 Properties

In TRVLabelItemInfo

Alignment 971
Cursor 971
MinWidth 971
ProtectTextStyleNo 972
RemoveInternalLeading 972
RemoveSpaceBelow 972
Text 972
TextStyleNo 973

7.2.1.1 TRVLabelItemInfo.Alignment

Defines a horizontal alignment of text relative to the item.

```
property Alignment: TAlignment;
```

This property makes sense if Text 972 width is less than MinWidth 971.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx 566 to change value of this property as an editing operation (*rveipcAlignment* 1119).

Default value:

taLeftJustify

7.2.1.2 TRVLabelItemInfo.Cursor

Cursor for the item.

```
Cursor: TCursor;
```

Value of this property is not stored in RVF 129.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx 566 to change value of this property as an editing operation (*rveipcCursor* 1119).

Default value:

crDefault (using default cursor)

7.2.1.3 TRVLabelItemInfo.MinWidth

Defines a minimal width of this item.

```
property MinWidth: TRVStyleLength 1094;
```

This value is measured in Units 764 of the linked TRVStyle 739 component.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx 566 to change value of this property as an editing operation (*rveipcMinWidth* 1119).

Default value:

0

See also properties:

- Alignment 971;
- Text 972.

7.2.1.4 TRVLabelItemInfo.ProtectTextStyleNo

If set to *True*, TCustomRichViewEdit.ApplyTextStyle[502] and ApplyStyleConversion[501] will not be able to change TextStyleNo[973] of this item.

```
ProtectTextStyleNo: Boolean;
```

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change value of this property as an editing operation (*rveipcProtectTextStyleNo*[1119]).

Default value:

False

7.2.1.5 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[972] contains accented characters.

If **RemoveSpaceBelow**=*True*, the label item is displayed without space below the text. This property is used only if **UseTypoMetric** property = *True*.

These properties are useful if this label item is used as a drop cap (the first letter in a chapter, with vertical alignment[1100] = *rvvaLeft*).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change values of these properties as an editing operation (*rveipcRemoveInternalLeading and rveipcRemoveSpaceBelow*[1119]).

Default value:

- RemoveInternalLeading: *False*
- RemoveSpaceBelow: *False*
- UseTypoMetric: *True*

7.2.1.6 TRVLabelItemInfo.Text

Visible text

```
Text: TRVUnicodeString[1099];
```

(changed in version 18)

Unlike text items[167], this is not the text returned by TCustomRichView.GetItemText[310].

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[567] to change value of this property as an editing operation (RVLabelItem.*rvespcText*[1124]).

See also properties:

- Alignment[971];
- TextStyleNo[973].

7.2.1.7 TRVLabelItemInfo.TextStyleNo

Defines text attributes.

```
property TextStyleNo: Integer;
```

This is the index in the TextStyles[764] collection of the TRVStyle[739] component linked to the TCustomRichView[216] component containing this item.

If this item is inserted in editor[469], value of this property may be changed when ApplyTextStyle[502] or ApplyStyleConversion[501] method is called. You can prevent this by assigning *True* to ProtectTextStyleNo[972] property.

If style templates[263] are used, the component may automatically change value of this property to provide a consistency of text and paragraph style templates (even if ProtectTextStyleNo[972]=*True*).

See also properties:

- Text[972].

7.2.2 Methods

In TRVLabelItemInfo

Create[973]
CreateEx[973]

7.2.2.1 TRVLabelItemInfo.Create

A constructor, creates a new TRVLabelItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], where you wish to insert this item.

This constructor creates a new label with empty Text[972] and TextStyleNo[973]=0.

There is more convenient constructor, CreateEx[973].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.2.2.2 TRVLabelItemInfo.CreateEx

A constructor, creates a new TRVLabelItemInfo object with the specified Text[972] and TextStyleNo[973].

```
constructor CreateEx(RVData: TPersistent;
  TextStyleNo: Integer; const Text: TRVUnicodeString[1099]);
```

(changed in version 18)

Parameters:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469] where you wish to insert this item.

TextStyleNo – value for TextStyleNo [973] property.

Text – value for Text [972] property.

See also methods:

- Create [973].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.3 Math expression - TRVMathItemInfo

TRVMathItemInfo is a class representing mathematical expressions [193] (equations, formulas) in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem [276]) or inserted into TRichViewEdit (see InsertItem [531]). Style [310] of this item type: *rvsMathEquation* (-210)

This item requires RAD Studio XE4 or newer. This item type is not available in Lazarus and FireMonkey versions.

Unit RVMathItem.

Syntax

```
TRVMathItemInfo = class(TRVRectItemInfo)
```

(introduced in version 17)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]
TRVNonTextItemInfo
TRVRectItemInfo

License

This item uses Adit Math Engine by Adit Software: ***aditmath.com***

There are two version of Adit Math Engine: free version and commercial version.

Free version

Units of the free version 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.

To use the free version, include RVBasicMathWrapper.pas in your project (for example, add in in "uses" of the main form unit).

Commercial version

To use the commercial version, include RVAdvMathWrapper.pas in your project (for example, add in in "uses" of the main form unit).

Properties

The expression is defined in LaTeX-like language; it must be assigned to Text [977] property.

By default, the item uses font name, size and color defined in TRVMathDocObject [1037] item of DocObjects [235] collection (or "Cambria Math, 10pt, *clWindowText*, if this object does not exist). You can override them in FontName [976], FontSizeDouble [976], TextColor [977] properties.

Use SetCurrentItemExtraIntPropertyEx [566] and SetCurrentItemExtraStrPropertyEx [567] to change properties of mathematical items as editing operations.

7.3.1 Properties

In TRVMathItemInfo

DisplayInline [975]
FontName [976]
FontSizeDouble [976]
Text [977]
TextColor [977]

7.3.1.1 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 [566] to change value of this property as an editing operation (*rveipcDisplayInline* [1119]).

Default value:

False

7.3.1.2 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[1037] item of DocObjects[235] 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[567] to change value of this property as an editing operation (*rvespcFontName*[1124]).

See also:

- FontSizeDouble[976]

7.3.1.3 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[1037] item of DocObjects[235] collection.

This value is measured in half-points, i.e. FontSizeDouble = 20 means 10pt, FontSizeDouble=21 means 10.5pt.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change value of this property as an editing operation (*rveipcFontSizeDouble*[1119]).

See also:

- FontName[976]

7.3.1.4 TRVMathItemInfo.Text

Mathematical expression in LaTeX-like format.

```
property Text: TRVUnicodeString[1099];
```

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[167], this is not the text returned by TCustomRichView.GetItemText[310].

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[567] to change value of this property as an editing operation (RVMathItem.*rvespcText*[1124]).

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}{\displaystyle a_3 + \frac{1}{a_4} } } }
```

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

If the expression is incorrect, the item is displayed in an erroneous state.

7.3.1.5 TRVMathItemInfo.TextColor

The text color.

```
property TextColor: TRColor;
```

The default value of this property is *clNone*. This means the item uses the default text color defined in TRVMathDocObject[696] item of DocObjects[235] collection.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change value of this property as an editing operation (*rveipcTextColor*[1119]).

Default value:

clNone

7.3.2 Methods

In TRVMathItemInfo

Create[978]

7.3.2.1 TRVMathItemInfo.Create

A constructor, creates a new TRVMathItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469], where you wish to insert this item.

This constructor creates a new item with empty Text [977].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.4 Numbered sequence - TRVSeqItemInfo

TRVSeqItemInfo is a class representing numbered sequence [183] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem [276]) or inserted into TRichViewEdit (see InsertItem [531]). Style [310] of this item type: *rvsSequence* (-202)

Unit RVSeqItem.

Syntax

```
TRVSeqItemInfo = class(TRVLabelItemInfo [970])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo [970]

Properties

The main property of this class is SeqName [981]. All items having the same value of this property are numbered continuously. You can restart numbering from the specified value, see StartFrom [981] and Reset [981] properties.

Numbering type is defined in NumberType [980] property.

Font for this item is defined in TextStyleNo [973]. ProtectTextStyleNo [972] can be used to protect TextStyleNo [973] from changing.

Text [972] for this item is calculated automatically.

Use SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change properties of label items as editing operations.

Related Properties of TRVStyle[739]

- FieldHighlightColor[746] – color for highlighting numbering sequences (overrides background color[808] specified in the text style)
- FieldHighlightType[747] specifies when to highlight numbering sequences.

Numbering Sequences and List Markers

Numbered sequences are similar to list markers[180]. 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[981] 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[755].
Items in the same sequence may have different fonts, numbering type, format strings.	All list markers linked to the same list level have the same font, numbering type and format string.
TCustomRichViewEdit.ApplyTextStyle[502] and ApplyStyleConversion[501] can change item font.	TCustomRichViewEdit.ApplyTextStyle[502] and ApplyStyleConversion[501] cannot change item font.

Inherited Classes

The following item types are inherited from this class:

- TRVEndNoteItemInfo[985] (endnote);
- TRVFootnoteItemInfo[988] (footnote);

See Also...

Demos:

- Demos*\Assorted\Fields\MailMerge-LabelItems\

7.4.1 Properties

In TRVSeqItemInfo

FormatString[980]

NumberType[980]
Reset[981]
SeqName[981]
StartFrom[981]

Inherited from TRVLabelItemInfo[970]

Alignment[971]
Cursor[971]
MinWidth[971]
ProtectTextStyleNo[972]
Text[972]
TextStyleNo[973]

7.4.1.1 TRVSeqItemInfo.FormatString

Text format.

```
FormatString: TRVUnicodeString[1099];
```

(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[980] = *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[340].

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[567] to change value of this property as an editing operation.

Default value:

'' (empty string)

7.4.1.2 TRVSeqItemInfo.NumberType

Numbering type.

```
property NumberType: TRVSeqType[1092];
```

See TRVSeqType[1092] for possible values.

This property is initialized in the constructor CreateEx[982].

This property is ignored in inherited classes: TRVFootnoteItemInfo[988], TRVEndnoteItemInfo[985], TRVSidenoteItemInfo[990], TRVTextBoxItemInfo[1004].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences[340].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change value of this property as an editing operation.

See also properties:

- FormatString [980].

7.4.1.3 TRVSeqItemInfo.Reset

Resets counter to StartFrom [981].

```
Reset: Boolean;
```

This property is initialized in constructor CreateEx [982].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences [340].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [566] to change value of this property as an editing operation.

7.4.1.4 TRVSeqItemInfo.SeqName

Name of sequence. Items having the same value of this property are numbered continuously.

```
SeqName: TRVUnicodeString [1099];
```

(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 [988]);
- '@endnote@' (used in TRVEndnoteItemInfo [985]).
- '@sidenote@' (used in TRVSidenoteItemInfo [990]).
- '@textbox@' (used in TRVTextBoxItemInfo [1004]).

This property is initialized in constructor CreateEx [982].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [566] to change value of this property as an editing operation.

7.4.1.5 TRVSeqItemInfo.StartFrom

Value of counter for this item, if Reset [981] = *True*.

```
StartFrom: Integer;
```

This property is initialized in constructor CreateEx [982].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences [340].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [566] to change value of this property as an editing operation.

7.4.2 Methods

In TRVSeqItemInfo

Create [982]
CreateEx [982]

7.4.2.1 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[982] instead.

7.4.2.2 TRVSeqItemInfo.CreateEx

A constructor, creates a new TRVSeqItemInfo object.

```
constructor CreateEx(RVData: TPersistent;
  const ASeqName: TRVUnicodeString[1099];
  ANumberType: TRVSeqType; ATextStyleNo, AStartFrom: Integer;
  AReset: Boolean);
```

(changed in version 18)

Parameters:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

ASeqName – value for SeqName[981] property.

ANumberType – value for NumberType[980] property.

ATextStyleNo – value for TextStyleNo[973] property.

AStartFrom – value for StartFrom[981] property.

AReset – value for Reset[981] property.

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.5 Ancestor for notes - TCustomRVNoteItemInfo

TCustomRVNoteItemInfo is an ancestor class for footnote[185], endnote[184], sidenote[186] and text box[188] items.

Objects of this class must not be created directly, create objects of the following inherited classes:

- TRVFootnoteItemInfo[988];
- TRVEndnoteItemInfo[985];
- TRVSidenoteItemInfo[990];
- TRVTextBoxItemInfo[1004].

Unit RVNote.

Syntax

```
TCustomRVNoteItemInfo = class(TRVSeqItemInfo[978])
```

```
TRVFootOrEndnoteItemInfo = class(TCustomRVNoteItemInfo)
```

(introduced in version 10 and 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo 899
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo 970
TRVSeqItemInfo 978

Properties

This class introduces Document 984 property – footnote/endnote/sidenote/textbox text. This text is not displayed in TCustomRichView, but it is displayed on printing and in ScaleRichView.

Document 984 uses the same TRVStyle component as the main document (where a note is inserted).

You can use ReplaceDocumentEd 985 method to change Document 982 as an editing operation.

See Also...

Demos:

- Demos*\Editors\Notes

7.5.1 Properties

In TCustomRVNoteItemInfo

- Document 984
- NoteText 984

Inherited from TRVSeqItemInfo 978

- FormatString 980
- NumberType 980
- Reset 981
- SeqName 981
- StartFrom 981

Inherited from TRVLabelItemInfo 970

- Alignment 971
- Cursor 971
- MinWidth 971
- ProtectTextStyleNo 972
- Text 972
- TextStyleNo 973

7.5.1.1 TCustomRVNoteItemInfo.Document

Footnote/endnote text (displayed on printing)

```
property Document: TRVNoteData[1014];
```

You can use ReplaceDocumentEd[985] method to change Document as an editing operation.

7.5.1.2 TCustomRVNoteItemInfo.NoteText

Text for assigning to NoteText[247] property of TRichViewEdit[469].

```
property NoteText: TRVUnicodeString[1099];
```

(introduced in version 15; changed in version 18)

This property returns Text[972].

If TRichViewEdit is used as an editor for Document[984], you should assign this property to its NoteText[247] property, to allow a proper displaying of note references[190].

7.5.2 Methods

In TCustomRVNoteItemInfo

CreateEx[984]
ReplaceDocumentEd[985]

7.5.2.1 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[988];
- TRVEndnoteItemInfo[985];
- TRVSidenoteItemInfo[990];
- TRVTextBoxItemInfo[1004].

Parameters

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[973] property. It's recommended to use RVGetNoteTextStyleNo[1052] function for assigning this parameter.

AStartFrom – value for StartFrom[981] property.

AReset – value for Reset[981] property.

7.5.2.2 TCustomRVNoteItemInfo.ReplaceDocumentEd

Replaces content of Document[984] as an editing operation.

```
procedure ReplaceDocumentEd(Stream: TStream);
```

Stream must contain new document in RVF format[129].

7.6 Endnote - TRVEndnoteItemInfo

TRVEndnoteItemInfo is a class representing an endnote[184] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[276]) or inserted into TRichViewEdit (see InsertItem[531]). Style[310] of this item type: *rvsEndnote* (-204)

Unit RVNote.

Syntax

```
TRVEndnoteItemInfo = class(TRVFootOrEndnoteItemInfo[982])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[970]
TRVSeqItemInfo[978]
TCustomRVNoteItemInfo[982]
TRVFootOrEndnoteItemInfo[982]

Using

Numbering type for endnotes is defined in EndnoteNumbering[746] property of TRVStyle component (NumberType[980] property, inherited from TRVSeqItemInfo[978], is ignored).

You can enumerate all endnotes in documents using RVGetFirstEndnote and RVGetNextEndnote[1052] functions.

Use RVGetNoteTextStyleNo[1052] to assign TextStyleNo[973] property (in the constructor).

Limitation: in TRVReportHelper[721], all pages consisting only of endnotes' texts have the same height as the last document's page.

Use SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change properties of label items as editing operations.

Example

This example inserts a new endnote in rveMain (TCustomRichViewEdit). TextStyleNo[973] for this endnote is returned by RVGetNoteTextStyleNo[1052] function basing on the rveMain.CurTextStyleNo[482] -th style).

Document[984] for this endnote contains one reference[190] (TextStyleNo[973] for this reference is returned by RVGetNoteTextStyleNo[1052] function basing on the 0th style) and one space character of the 0th style.

```
var EndNote: TRVEndnoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo[1006];
begin
  EndNote := TRVEndnoteItemInfo.CreateEx[987](rveMain.RVData[254],
    RVGetNoteTextStyleNo[1052](rvs, rveMain.CurTextStyleNo[482]), 1, False);
  NoteRef := TRVNoteReferenceItemInfo[1006].CreateEx[1008](EndNote.Document[984],
    RVGetNoteTextStyleNo[1052](rveMain.Style[260],0));
  EndNote.Document[984].AddItem[276]('', NoteRef);
  EndNote.Document[984].AddNL[277](' ', 0, -1);
  if rveMain.InsertItem[531]('', EndNote) then
    ...
end;
```

The topic about TRVNoteReferenceItemInfo[1006] contains example showing how to edit Document[984].

See Also...

Demos:

- Demos*\Editors\Notes

7.6.1 Properties

Inherited from TCustomRVNoteItemInfo[982]

- Document[984]
- NoteText[984]

Inherited from TRVSeqItemInfo[978]

- FormatString[980]
- NumberType[980]
- Reset[981]
- SeqName[981]
- StartFrom[981]

Inherited from TRVLabelItemInfo[970]

- Alignment[971]
- Cursor[971]
- MinWidth[971]
- ProtectTextStyleNo[972]
- Text[972]
- TextStyleNo[973]

7.6.2 Methods

In TRVEndnoteItemInfo

- Create[987]

CreateEx[987]

Inherited from TCustomRVNoteItemInfo[982]

ReplaceDocumentEd[985]

7.6.2.1 TRVEndnoteItemInfo.Create

A constructor, creates a new TRVEndnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], where you wish to insert this item.

This constructor creates a new endnote with TextStyleNo[973]=0.

There is more convenient constructor, CreateEx[987].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.6.2.2 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[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[973] property. It's recommended to use RVGetNoteTextStyleNo[1052] function for assigning this parameter.

AStartFrom – value for StartFrom[981] property.

AReset – value for Reset[981] property.

See also methods:

- Create[987].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.7 Footnote - TRVFootnoteItemInfo

TRVFootnoteItemInfo is a class representing a footnote[185] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[276]) or inserted into TRichViewEdit (see InsertItem[531]). Style[310] of this item type: *rvsFootnote* (-203)

Unit RVNote.

Syntax

```
TRVFootnoteItemInfo = class(TRVFootOrEndnoteItemInfo[982])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[970]
TRVSeqItemInfo[978]
TCustomRVNoteItemInfo[982]
TRVFootOrEndnoteItemInfo[982]

Using

Numbering type for footnotes is defined in FootnoteNumbering[746] property of TRVStyle[739] component (NumberType[980] property, inherited from TRVSeqItemInfo[978], is ignored).

You can enumerate all footnotes in documents using RVGetFirstFootnote and RVGetNextFootnote[1052] functions.

Use RVGetNoteTextStyleNo[1052] to assign TextStyleNo[973] property (in the constructor).

Footnotes are numbered continuously in TCustomRichView. But when printing, footnote numbering can be restarted on each page, see FootnotePageReset[749] property of TRVStyle[739] component.

Limitation: footnote's Document[984] must not be higher than the page height. Unlike endnotes[184], footnote is printed on a single page.

Use SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change properties of label items as editing operations.

Example

This example inserts a new footnote in rveMain (TCustomRichViewEdit). TextStyleNo[973] for this footnote is returned by RVGetNoteTextStyleNo[1052] function basing on the rveMain.CurTextStyleNo[482] -th style).

Document[984] for this footnote contains one reference[190] (TextStyleNo[973] for this reference is returned by RVGetNoteTextStyleNo[1052] function basing on the 0th style) and one space character of the 0th style.

```
var FootNote: TRVFootnoteItemInfo;
```

```
    NoteRef: TRVNoteReferenceItemInfo[1006];
begin
  FootNote := TRVFootnoteItemInfo.CreateEx[990](rveMain.RVData[254],
    RVGetNoteTextStyleNo[1052](rvs, rveMain.CurTextStyleNo[482]), 1, False);
  NoteRef := TRVNoteReferenceItemInfo[1006].CreateEx[1008](FootNote.Document[984],
    RVGetNoteTextStyleNo[1052](rveMain.Style[260],0));
  FootNote.Document[984].AddItem[276]('', NoteRef);
  FootNote.Document[984].AddNL[277](' ', 0, -1);
  if rveMain.InsertItem[531]('', FootNote) then
    ...
end;
```

The topic about TRVNoteReferenceItemInfo[1006] contains example showing how to edit Document[984].

See Also...

Demos:

- Demos*\Editors\Notes

7.7.1 Properties

Inherited from TCustomRVNoteItemInfo[982]

- Document[984]
- NoteText[984]

Inherited from TRVSeqItemInfo[978]

- FormatString[980]
- NumberType[980]
- Reset[981]
- SeqName[981]
- StartFrom[981]

Inherited from TRVLabelItemInfo[970]

- Alignment[971]
- Cursor[971]
- MinWidth[971]
- ProtectTextStyleNo[972]
- Text[972]
- TextStyleNo[973]

7.7.2 Methods

In TRVFootnoteItemInfo

- Create[990]
- CreateEx[990]

Inherited from TCustomRVNoteItemInfo[982]

- ReplaceDocumentEd[985]

7.7.2.1 TRVFootnoteItemInfo.Create

A constructor, creates a new TRVFootnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], where you wish to insert this item.

This constructor creates a new footnote with TextStyleNo[973]=0.

There is more convenient constructor, CreateEx[990].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.7.2.2 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[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[973] property. It's recommended to use RVGetNoteTextStyleNo[1052] function for assigning this parameter.

AStartFrom – value for StartFrom[981] property.

AReset – value for Reset[981] property.

See also methods:

- Create[990].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.8 Sidenote - TRVSidenoteItemInfo

TRVSidenoteItemInfo is a class representing a sidenote[186] 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[276]) or inserted into TRichViewEdit (see InsertItem[531]). Style[310] of this item type: *rvsSidenote* (-206)

Unit RVSidenote.

Syntax

```
TRVSidenoteItemInfo = class(TCustomRVNoteItemInfo[982])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[970]
TRVSeqItemInfo[978]
TCustomRVNoteItemInfo[982]

Using

Numbering type for sidenotes is defined in SidenoteNumbering[746] property of TRVStyle[739] component (NumberType[980] property, inherited from TRVSeqItemInfo[978], is ignored).

You can enumerate all sidenotes in documents using RVGetFirstSidenote and RVGetNextSidenote[1052] functions (these functions return text box items[188] as well)

Use RVGetNoteTextStyleNo[1052] to assign TextStyleNo[973] property (in the constructor).

This class introduces the following properties:

- BoxPosition[993] – contains properties defining the position of the floating box
- BoxProperties[992] – contains properties describing size, background and border of the floating box.

Use SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change properties of sidenotes as editing operations.

Example

This example inserts a new sidenote in rveMain (TCustomRichViewEdit). TextStyleNo[973] for this sidenote is returned by RVGetNoteTextStyleNo[1052] function basing on the rveMain.CurTextStyleNo[482]-th style).

Document[984] for this sidenote contains one reference[190] (TextStyleNo[973] for this reference is returned by RVGetNoteTextStyleNo[1052] 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[1006];
begin
  SideNote := TRVSidenoteItemInfo.CreateEx[990](rveMain.RVData[254],
    RVGetNoteTextStyleNo[1052](rvs, rveMain.CurTextStyleNo[482]), 1, False);
  NoteRef := TRVNoteReferenceItemInfo[1006].CreateEx[1008](SideNote.Document[984],
    RVGetNoteTextStyleNo[1052](rveMain.Style[260],0));
```

```
  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 contains example showing how to edit Document.

See also

- TRVTexBoxItemInfo

7.8.1 Properties

In TRVSidenoteItemInfo

BoxPosition
BoxProperties

Inherited from TCustomRVNoteItemInfo

- Document
- NoteText

Inherited from TRVSeqItemInfo

FormatString
NumberType
Reset
SeqName
StartFrom

Inherited from TRVLabelItemInfo

Alignment
Cursor
MinWidth
ProtectTextStyleNo
Text
TextStyleNo

7.8.1.1 TRVSidenoteItemInfo.BoxProperties

Properties of a floating box: size, border, background, vertical alignment of content.

```
property BoxProperties: TRVBoxProperties;
```

7.8.1.2 TRVSidenoteItemInfo.BoxPosition

Properties of a floating box: horizontal and vertical positions, z-order.

```
property BoxPosition: TRVBoxPosition[997];
```

7.8.2 Methods

In TRVSidenoteItemInfo

Create[993]
CreateEx[993]

Inherited from TCustomRVNoteItemInfo[982]

ReplaceDocumentEd[985]

7.8.2.1 TRVSidenoteItemInfo.Create

A constructor, creates a new TRVSidenoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], where you wish to insert this item.

This constructor creates a new sidenote with TextStyleNo[973]=0.

There is more convenient constructor, CreateEx[993].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.8.2.2 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[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[973] property. It's recommended to use RVGetNoteTextStyleNo[1052] function for assigning this parameter.

AStartFrom – value for StartFrom[981] property.

AReset – value for Reset[981] property.

See also methods:

- Create [993].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.8.3 Classes of properties

Classes of TRVSidenoteItemInfo [990] Properties

- TRVBoxProperties [994] defines the text box size and appearance. This is a type of BoxProperties [992] property.
- TRVBoxPosition [997] defines the text box position. This is a type of BoxPosition [993] property.

7.8.3.1 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 [992] property of sidenotes [186] and text boxes [188].

Properties:

- Background [995];
- Border [995];
- Height, HeightType [995];
- Width, WidthType [996];
- VAlign [996].

7.8.3.1.1 Properties

In TRVBoxProperties

Background [995]
Border [995]
Height [995]
HeightType [995]
Width [996]
WidthType [996]
VAlign [996]

7.8.3.1.1.1 TRVBoxProperties.Background

Defines background properties.

```
type
  TRVBoxBackgroundRect = class (TRVBackgroundRect[851])
property Background: TRVBoxBackgroundRect;
```

TRVBoxBackgroundRect changes the initial values of properties introduced in TRVBackgroundRect[851]:

- Color[852]=*clWindow*
- BorderOffsets[851]: Left=Right=10, Top=Bottom=5.

Unlike in TRVBackgroundRect, in TRVBoxBackgroundRect negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change value of subproperties of this property as an editing operation.

7.8.3.1.1.2 TRVBoxProperties.Border

Defines background properties.

```
type
  TRVBoxBorder = class (TRVBorder[852])
property Border: TRVBoxBorder;
```

TRVBoxBorder changes the initial values of properties introduced in TRVBorder[852]:

- Style[854]=*rvbSingle*
- BorderOffsets[853]: Left=Right=10, Top=Bottom=5.

Unlike in TRVBorder, in TRVBoxBorder negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change values of subproperties of this property as an editing operation.

7.8.3.1.1.3 TRVBoxProperties.Height, HeightType

These properties define a height of a floating box.

```
type
  TRVBoxHeightType = (rvbhtAbsolute, rvbhtRelPage, rvbhtRelMainTextArea,
    rvbhtRelTopMargin, rvbhtRelBottomMargin, rvbhtAuto);
property HeightType: TRVBoxHeightType;
property Height: TRVFloatSize[1074];
```

Value of HeightType	Meaning
rvbhtAbsolute	**Height** specifies the box height in Units[764] of TRVStyle
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

Value of HeightType	Meaning
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[566] and SetCurrentItemExtraStrPropertyEx[567] 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

7.8.3.1.1.4 TRVBoxProperties.VAlign

Specifies a vertical alignment of a box content inside a box.

```
property VAlign: TTextLayout;
```

This property makes sense if HeightType[995]<>*rvbhtAuto.*

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change value of this property as an editing operation.

Default value:

tlTop

7.8.3.1.1.5 TRVBoxProperties.Width, WidthType

These properties define a width of a floating box.

```
type
  TRVBoxWidthType = (rvbwtAbsolute, rvbwtRelPage, rvbwtRelMainTextArea,
    rvbwtRelLeftMargin, rvbwtRelRightMargin,
    rvbwtRelInnerMargin, rvbwtRelOuterMargin);
```

```
property WidthType: TRVBoxWidthType;
property Width: TRVFloatSize[1074];
```

Value of WidthType	Meaning
rvbwtAbsolute	**Width** specifies the box width in Units[764] of TRVStyle
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[566] and SetCurrentItemExtraStrPropertyEx[567] 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

7.8.3.2 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 [993] property of sidenotes [186] and text boxes [188].

Properties:

- HorizontalAnchor [998] and HorizontalPositionKind, HorizontalOffset, HorizontalAlignment [999];
- VerticalAnchor [1002] and VerticalPositionKind, VerticalOffset, VerticalAlignment [1003];
- RelativeToCell [1002];
- PositionInText [1001].

7.8.3.2.1 Properties

In TRVBoxPosition

HorizontalAlignment [999]
HorizontalAnchor [998]
HorizontalOffset [999]
HorizontalPositionKind [999]
PositionInText [1001]
RelativeToCell [1002]
VerticalAlignment [1003]
VerticalAnchor [1002]
VerticalOffset [1003]
VerticalPositionKind [1003]

7.8.3.2.1.1 TRVBoxPosition.HorizontalAnchor

Specifies a horizontal anchor area for a floating box.

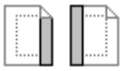
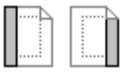
```
type
  TRVHorizontalAnchor = (rvhanCharacter, rvhanPage, rvhanMainTextArea,
    rvhanLeftMargin, rvhanRightMargin, rvhanInnerMargin, rvhanOuterMargin);
property HorizontalAnchor: TRVHorizontalAnchor;
```

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 [696], a left margin starts from the page's left edge and ends at Margins [705].Left.

If RelativeToCell [1002]=*True*, and a sidenote [186] (or a text box item [188]) is inserted in a table cell [198], then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
rvhanCharacter	A sidenote [186] character, or a place of insertion of a text box item [188].	

Value	Anchor Area	Scheme
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[566] and SetCurrentItemExtraStrPropertyEx[567] 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

7.8.3.2.1.2 TRVBoxPosition.HorizontalXXX

These properties define a horizontal position of a floating box.

```
type
  TRVFloatHorizontalAlignment = (rvfphalLeft, rvfphalCenter, rvfphalRight);
property HorizontalPositionKind: TRVFloatPositionKind[1073];
property HorizontalOffset: TRVFloatPosition[1073];
property HorizontalAlignment: TRVFloatHorizontalAlignment;
```

A position is calculated relative to the area specified in HorizontalAnchor[998].

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 [998].

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 Units [764] of the linked TRVStyle [739].

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%	

Value of HorizontalOffset	Meaning	Scheme
100000	100%	
50000	50%	
-100000	-100%	

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] and SetCurrentItemExtraStrPropertyEx[567] to change values of these properties as an editing operation.

Default values:

- HorizontalPositionKind: *rvfpkAbsPosition*
- HorizontalOffset: 0
- HorizontalAlignment: *rvfphalLeft*

7.8.3.2.1.3 TRVBoxPosition.PositionInText

Specifies how the text box is positioned relative to a text where a sidenote[186] (or a text box item[188]) 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[566] and SetCurrentItemExtraStrPropertyEx[567] to change value of this property as an editing operation.

Default value:

rvpitAboveText

7.8.3.2.1.4 TRVBoxPosition.RelativeToCell

Allows anchoring a text box to a table cell [198].

```
property RelativeToCell: Boolean;
```

If *True*, and a sidenote [186] (or a text box item [188]) is placed in a table cell, then horizontal [998] and vertical [1002] positions are calculated relative to this cell (instead of a page).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [566] and SetCurrentItemExtraStrPropertyEx [567] to change value of this property as an editing operation.

Default value:

True

7.8.3.2.1.5 TRVBoxPosition.VerticalAnchor

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 [696], a top margin starts from the page's top edge and ends at Margins [705].Top.

If RelativeToCell [1002]=*True*, and a sidenote [186] (or a text box item [188]) is inserted in a table cell [198], then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
rvvanLine	A line where a sidenote [186] character (or a text box item [188]) is inserted.	
rvvanParagraph	A paragraph where a sidenote [186] character (or a text box item [188]) 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 [566] and SetCurrentItemExtraStrPropertyEx [567] 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

7.8.3.2.1.6 TRVBoxPosition.VerticalXXX

These properties define a vertical position of a floating box.

```
type
  TRVFloatVerticalAlignment = (rvfpvalTop, rvfpvalCenter, rvfpvalBottom);
property VerticalPositionKind: TRVFloatPositionKind[1073];
property VerticalOffset: TRVFloatPosition[1073];
property VerticalAlignment: TRVFloatVerticalAlignment;
```

A position is calculated relative to the area specified in VerticalAnchor[1002].

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[1002].

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 Units 764 of the linked TRVStyle 739.

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 566 and SetCurrentItemExtraStrPropertyEx 567 to change values of these properties as an editing operation.

Default values:

- VerticalPositionKind: *rvfpkAbsPosition*
- VerticalOffset: 0
- VerticalAlignment: *rvfpvalTop*

7.9 Text box - TRVTextBoxItemInfo

TRVTextBoxItemInfo is a class representing text box item 188 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 276) or inserted into TRichViewEdit (see InsertItem 531). Style 310 of this item type: *rvsTextBox* (-207)

Unit RVSidenote.

Syntax

```
TRVTextBoxItemInfo = class(TRVSidenoteItemInfo 990)
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo 899
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo 970
TRVSeqItemInfo 978
TCustomRVNoteItemInfo 982
TRVSidenoteItemInfo 990

Using

This is a simplified version of a sidenote [186]. It has all properties of a sidenote, but the most of them are ignored, except for BoxPosition [993] and BoxProperties [992].

A special mark can be drawn in places of insertion of text box items, see TRVStyle.FloatingLineColor [747].

7.9.1 Methods

In TRVSidenoteItemInfo

Create [1005]
CreateEx [1005]

Inherited from TCustomRVNoteItemInfo [982]

ReplaceDocumentEd [985]

7.9.1.1 TRVTextBoxItemInfo.Create

A constructor, creates a new TRVFootnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469], where you wish to insert this item.

This constructor creates a new text box item with TextStyleNo [973]=0.

See also CreateEx [1005].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.9.1.2 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 [1005].

Parameters:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469] where you wish to insert this item.

ATextStyleNo – ignored.

AStartFrom – value for StartFrom [981] property. Assigned, but not used in this type of item.

AReset – value for Reset [981] property. Assigned, but not used in this type of item.

See also methods:

- Create [1005].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.10 Reference to the parent note - TRVNoteReferenceItemInfo

TRVNoteReferenceItemInfo is an item type referring to [190] the parent footnote [185], endnote [184], or sidenote [186] item. Objects of this class are usually appended to the note's Document [984] (see AddItem [276]). Besides, they can be appended to TRichView (see AddItem [276]) or inserted into TRichViewEdit (see InsertItem [531]). Style [310] of this item type: *rvsNoteReference* (-205).

Unit RVNote.

Syntax

```
TRVNoteReferenceItemInfo = class(TRVLabelItemInfo [970])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo [970]

Using

When inserted in a note's Document [984], the note reference displays the same text as the parent note.

When inserted in TCustomRichView, it displays NoteText [247].

Text [972] for this item is calculated automatically.

Use RVGetNoteTextStyleNo [1052] to assign TextStyleNo [973] property (in the constructor).

Use SetCurrentItemExtraIntPropertyEx [566] and SetCurrentItemExtraStrPropertyEx [567] to change properties of label items as editing operations.

Example

See examples in TRVFootnoteItemInfo [988] and TRVEndNoteItemInfo [985] showing how to insert footnote/endnote with document containing note reference.

This example shows how to edit note document in rveNote (TCustomRichViewEdit[469]). The footnote/endnote is NoteItem.

This code copies document from footnote/endnote to the editor:

```
var Stream: TMemoryStream;
begin

  rveNote.NoteText[247] := NoteItem.Text[972];
  rveNote.Clear[286];
  Stream := TMemoryStream.Create;
  try
    NoteItem.Document[984].SaveRVFToStream[352](Stream);
    Stream.Position := 0;
    rveNote.LoadRVFFromStream[329](Stream);
  finally
    Stream.Free;
  end;
  with rveNote do begin
    Format[295];
    SetSelectionBounds[372](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[247]. 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[487] then
    exit;
  Stream := TMemoryStream.Create;
  try
    rveNote.SaveRVFToStream[352](Stream, False);
    NoteItem.ReplaceDocumentEd[985](Stream);
  finally
    Stream.Free;
  end;
  rveNote.Modified[487] := False;
```

See Also...

Demos:

- Demos*\Editors\Notes

7.10.1 Properties

Inherited from TRVLabelItemInfo[970]

Alignment[971]
Cursor[971]
MinWidth[971]
ProtectTextStyleNo[972]
Text[972]

TextStyleNo[973]

7.10.2 Methods

In TRVNoteReferenceItemInfo

Create[1008]
CreateEx[1008]

7.10.2.1 TRVNoteReferenceItemInfo.Create

A constructor creating a new TRVNoteReferenceItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469], where you wish to insert this item.

This constructor creates a new note reference with TextStyleNo[973]=0.

There is more convenient constructor, CreateEx[1008].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.10.2.2 TRVNoteReferenceItemInfo.CreateEx

A constructor creating a new TRVNoteReferenceItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; TextStyleNo: Integer);
```

Parameters:

RVData – RVData[254] property of TRichView[216] or TRichViewEdit[469] where you wish to insert this item.

TextStyleNo – value for TextStyleNo[973] property. It's recommended to use RVGetNoteTextStyleNo[1052] function for assigning this parameter.

See also methods:

- Create[1008].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

7.11 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[1010];
- TRVPageCountItemInfo[1011].

Unit RVFieldItems.

Syntax

```
TCustomRVFieldItemInfo = class (TRVLabelItemInfo[970]);
TCustomRVPageNumberItemInfo = class(TCustomRVFieldItemInfo);
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[970]

TCustomRVPageNumberItemInfo introduces a new property: NumberType[1009].

7.11.1 Properties

In TCustomRVPageNumberItemInfo

NumberType[1009]

Inherited from TRVLabelItemInfo[970]

Alignment[971]
Cursor[971]
MinWidth[971]
ProtectTextStyleNo[972]
Text[972]
TextStyleNo[973]

7.11.1.1 TCustomRVPageNumberItemInfo.NumberType

Specifies how the page number (or page count) is displayed.

```
NumberType: TRVPageNumberType[1082];
```

This property is initialized in the constructor.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[566] to change value of this property as an editing operation.

Default value:
rvpntDefault

7.12 Page Number - TRVPageNumberItemInfo

TRVPageNumberItemInfo displays the current page number[190].

Unit RVFieldItems.

Syntax

```
TRVPageNumberItemInfo = class(TCustomRVPageNumberItemInfo[1009])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[899]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[970]

Using

When inserted in TRichView[216], it is displayed as Text[972] (by default, as '{p}').

When printed with TRVPrint[696], it displays a number of the page where this item is located. Page numbers are counted from TRVPrint.PageNoFromNumber[881].

When drawing with TRVReportHelper[721], the default behavior is like in TRVPrint, but you can provide a value for the page number in OnGetPageNumber[728] 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[1009] property.

7.12.1 Methods

In TRVPageNumberItemInfo

Create[1010]
CreateEx[1011]

7.12.1.1 TRVPageNumberItemInfo.Create

A constructor creating a new TRVPageNumberItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469], where you wish to insert this item.

This constructor creates a new page number field with TextStyleNo [973]=0, Text [972]='{p}'.

There is more convenient constructor, CreateEx [1011].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.12.1.2 TRVPageNumberItemInfo.CreateEx

A constructor creating a new TRVPageNumberItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType[1082];
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469] where you wish to insert this item.

NumberType – value for NumberType [1009] property.

TextStyleNo – value for TextStyleNo [973] property.

This constructor creates a new page number field with Text [972]='{p}'.

See also methods:

- Create [1010].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.13 Page Count - TRVPageCountItemInfo

TRVPageCountItemInfo displays the count of pages [192].

Unit RVFieldItems.

Syntax

```
TRVPageCountItemInfo = class(TCustomRVPageNumberItemInfo[1009])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [899]

TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo [970]

Using

When inserted in TRichView [216], it is displayed as Text [972] (by default, as '{P}').

When printed with TRVPrint [696], 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 [1009] property.

7.13.1 Methods

In TRVPageCountItemInfo

Create [1012]
CreateEx [1012]

7.13.1.1 TRVPageCountItemInfo.Create

A constructor creating a new TRVPageCountItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469], where you wish to insert this item.

This constructor creates a new page count field with TextStyleNo [973]=0, Text [972]='{P}'.

There is more convenient constructor, CreateEx [1012].

See also methods of TCustomRichView:

- AddItem [276].

See also methods of TCustomRichViewEdit:

- InsertItem [531].

7.13.1.2 TRVPageCountItemInfo.CreateEx

A constructor creating a new TRVPageCountItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType [1082];
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData [254] property of TRichView [216] or TRichViewEdit [469] where you wish to insert this item.

NumberType – value for NumberType [1009] property.

TextStyleNo – value for TextStyleNo [973] property.

This constructor creates a new page count field with Text[972]='{P}'.

See also methods:

- Create[1012].

See also methods of TCustomRichView:

- AddItem[276].

See also methods of TCustomRichViewEdit:

- InsertItem[531].

8 Other classes

Classes providing uniform VCL and FireMonkey programming interface

- TRVCanvas[1018] – TCanvas wrapper.
- TRVGraphic[1032] – [VCL/LCL] TGraphic; [FMX] TBitmap wrapper.
- TRVPicture[1037] – [VCL/LCL] TPicture; [FMX] TPicture analog.
- TRVGraphicClass[1033] – [VCL/LCL] TGraphicClass; [FMX] class of TBitmap.

Classes for singleton objects

- TRVControlsPainter[1019] define appearance of controls. This class is not used in TRichView itself yet, but it is used in ScaleRichView and SRVControls: it defines appearance of controls, if their property SRVControlStyle = *srvcsSimple*.
- TRVCustomMathEquationManager[1020] [VCL] defines global properties for math expression items[193] in all documents.
- TRVDefaultLoadProperties[1026] defines default properties for loading HTML, Markdown, RTF, DocX documents.
- TRVGraphicHandler[1033] contains methods for working with graphics.
- TRVItemFactory[1036] creates items for documents.
- TRVThumbnailMaker[1041] provides high-quality image scaling in TRichView.

Other TRichView classes

- ERichViewError[1014] – an exception class.
- TCustomRVData[1014] – TRichView documents.
- TPrintableRV[1018] – a TRichView object used internally by TRVPrint[696].
- TRVBooleanRect[1018] – (Left, Top, Right, Bottom) boolean properties.
- TRVDeleteUnusedStylesData[1031] allows deleting unused styles in multiple documents.
- TRVIntegerList[1036] – a list of integer values.
- TRVMathDocObject[1037] contains properties common for all equation objects[193] in a document.

- TRVRect[1038] – (Left, Top, Right, Bottom) TRVStyleLength[1094] properties.
- TRVReportHelperWithHeaderFooters[1038] contains several TRVReportHelper[721] components representing a main document, headers and footers.
- TRVStringList[1041] – a string list that stores Unicode strings
- TRVUnitsRect[1042] – (Left, Top, Right, Bottom) TRVLength[1081] properties.

8.1 ERichViewError

This is an exception class used by TRichView Family of components. It does not introduce new methods and properties.

Unit [VCL/FMX]: RichView / fmxRichView;

Syntax

```
ERichViewError = class(Exception)
```

Hierarchy

TObject
Exception

8.2 TCustomRVData and others (TRichView documents)

These classes represent RichView documents. Documents contain a list of items. Some items may, in each order, contain subdocuments.

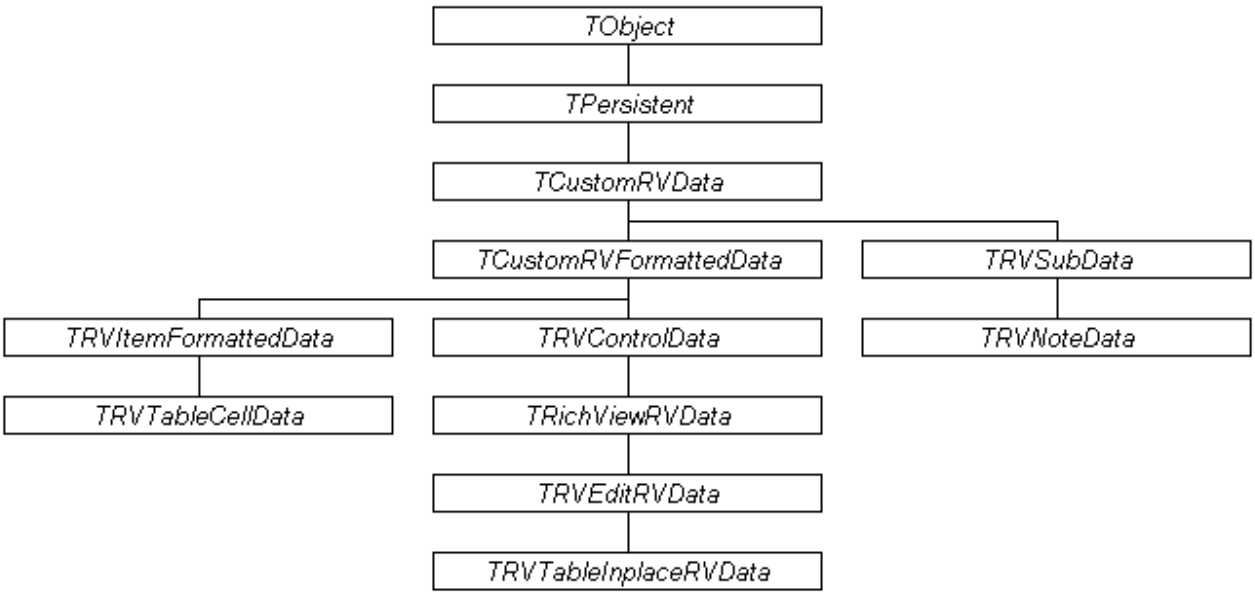
Class	Unit*	Meaning
TCustomRVData	CRVData	Basic class for TRichView documents.
TCustomRVFormattedData	CRVFData	Basic class for formatted TRichView documents (allows storing items coordinates).
TRVControlData	RVCtrlData	Basic class for documents owned by windowed control.
TRichViewRVData	RVRVData	Document in TRichView[216] and TDBRichView[604], a class of RVData[254] property of these components.
TRVEditRVData	RVERVData	Document in TRichViewEdit[469] and TDBRichViewEdit[616], a class of RVData[254] property of these components.
TRVTInplaceRVData	RVTInplace	Document in cell inplace editor, a class of of RVData[254] property of cell inplace editors.
TRVItemFormattedData	RVDataList	Basic class for formatted subdocuments owned by items.

Class	Unit*	Meaning
TRVTableCellData [951]	RVTable	Table [195] cell.
TRVSubData	RVSubData	Basic class for invisible documents owned by items.
TRVNoteData	RVNote	Document in footnote or endnote.

* these unit names are for VCL and LCL; for FireMonkey, unit names starts from "fmx", for example "fmxCRVData" instead of "CRVData".

This list does not include document classes used for printing.

Hierarchy



Overview

Special methods for **TCustomRVData**:

- Edit [1016];
- GetRVData [1017];
- GetSourceRVData [1017].

Special methods for **TCustomRVFormattedData**:

- GetOriginEx [1018].

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 [269],
- AddBullet [270],

- AddControl[272],
- AddHotspot[275],
- AddFmt[273], AddItem[276],
- AddNL[277], AddTextNL[281],
- AddPicture[278].

Obtaining information from cell:

- GetItemStyle[310], GetItemText -A -W[310],
- GetBreakInfo[296], GetBulletInfo[297], GetControlInfo[301], GetHotspotInfo[303], GetPictureInfo[318]
- GetItemTag[310], GetItemVAlign[311], IsParaStart[325], GetItemPara[309], IsFromNewLine[325]

Modifying items:

- SetBreakInfo[358], SetBulletInfo[359], SetControlInfo[361], SetHotspotInfo[364], SetPictureInfo[370], SetItemTag[367], SetItemVAlign[369], SetItemText -A -W[368]

See also

- Controls, Documents, Items[141]

8.2.1 TCustomRVData.Edit

Initializes editor for this RVData.

```
function Edit: TCustomRVData;
```

For RVData[254] of editor (main editor, or cell inplace editor), this method does nothing.

For table cell[951], this method creates cell inplace editor and returns its RVData[254].

If you want to select part of document in table cell, call Cell.Edit (or table.EditCell[931]) before calling cell.SetSelectionBounds[372] or cell.SelectAll[355].

This method must be called when the document is formatted.

Return value

GetRVData[1017] of the given RVData after initialization of editing.

Example

This example deletes all controls from rve: TRichViewEdit[469] 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[216]. TCustomRVFormattedData has all these methods and properties, with the same meaning.

```
procedure DeleteAllControls(RVData: TCustomRVData[1014];
  rve: TCustomRichViewEdit[469]);
var i,r,c: Integer;
    table: TRVTableItemInfo[901];
begin
  i := 0;
  while i<RVData.ItemCount[244] do begin
    case RVData.GetItemStyle[310](i) of
    rvsComponent[1127]:
      begin
```

```
        RVData := RVData.Edit;
        TCustomRVFormattedData(RVData).SetSelectionBounds(i, 0, i, 1);
        rve.DeleteSelection;
        i := rve.TopLevelEditor.CurItemNo-1;
      end;
    rvsTable:
      begin
        table := TRVTableItemInfo(RVData.GetItem(i));
        for r := 0 to table.RowCount-1 do
          for c := 0 to table.ColCount-1 do
            if table.Cells[r,c]<>nil then
              DeleteAllControls(table.Cells[r,c].GetRVData, rve);
      end;
    end; // of case
    inc(i);
  end; // of while
end;
```

Call:

```
DeleteAllControls(rve.RVData, rve);
```

8.2.2 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). When the cell is not edited, Cell.GetRVData = Cell. But when the cell is edited, this method returns RVData of the cell inplace editor.

Using RVData.GetRVData instead of RVData you can always access the document's items.

See also

- GetSourceRVData;
- Edit.

8.2.3 TCustomRVData.GetSourceRVData

This is an inverse method to GetRVData.

```
function GetSourceRVData: TCustomRVData;
```

For the most of document classes, this method returns Self, i.e. RVData.GetSourceRVData = RVData.

The only exception is RVData of the cell inplace editor, it returns the source cell (TRVTableCellData) for this editor.

See also

- GetRVData.

8.2.4 TCustomRVFormattedData.GetOriginEx

Returns coordinates of the given subdocument relative to the main document.

```
procedure GetOriginEx(out ALeft, ATop: TRVCoord[1064]);
```

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[306] and RVData.GetOriginEx.

This method does not take cell rotation into account, the coordinates are returned as if none of cells are rotated.

8.3 TPrintableRV

TPrintableRV is used internally by TRVPrint

8.4 TRVBooleanRect

This class contains 4 boolean properties: Left, Top, Right, Bottom

Unit [VCL/FMX] RVStyle / fmxRVStyle;

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[854];
- TRVTableCellData.VisibleBorders[963];
- TRVTableItemInfo.VisibleBorders[921].

8.5 TRVCanvas

A class that wraps TCanvas and provides additional methods for drawing.

Unit [VCL/FMX]: RVType / fmxRVType;

Syntax [VCL]:

```
TRVCanvas = Pointer;
```

Syntax [FMX]:

```
TRVCanvas = TRVFMXCanvas;
TRVFMXCanvas = class;
// TRVFMXCanvas is defined in fmxRVCanvasFM unit
```

Description [VCL]

In VCL, this class is used internally. Normally, this is a TCanvas class typecasted to Pointer.

Description [FireMonkey]

In FireMonkey, this class is not only used internally. It is also used as a type of parameters of several custom drawing events.

The main property of this class is Canvas: TCanvas.

8.6 TRVControlsPainter

This class contains properties and functions for drawing various standard controls (like buttons, scrollbars, combo boxes, etc.)

Unit RVControls;

Syntax

```
TRVControlsPainter = class;
```

(Introduced in version 18)

A single variable of this class is created: RVControlsPainter[1119]. You can access its properties.

Currently, this variable is not used in TRichView itself. However, it is used in ScaleRichView and (especially) in SRVControls. It used for controls having their property SRVControlStyle = srvcsSimple.

Color themes

There are several color themes for drawing controls. A theme can be applied by assigning a new value to **Theme**: TRVControlTheme property. TRVControlTheme is defined as:

```
type
  TRVControlTheme = (rvctPaleBlue, rvctSpringGreen,
    rvctSienna, rvctHighContrast);
```

The default value of this property is *rvctPaleBlue*.

When you apply a color theme, color properties of RVControlsPainter are changed according to this theme.

Colors

Alternatively, you can assign specific color properties:

BorderColor, HoverBorderColor, FocusedBorderColor, DisabledBorderColor – border around controls;

DisabledBackgroundColor – background of disabled controls;

ScrollButtonColor, HoverScrollButtonColor, PushedScrollButtonColor – scroll-bar buttons;

ScrollBarColor – scrollbar colorl

ScrollThumbColor, HoverScrollThumbColor, PushedScrollThumbColor – scroll-bar thumb;

ButtonBackgroundColor, HoverButtonBackgroundColor – push-button background;

ButtonBorderColor – push-button background;

PushedButtonTextColor – pushed push-button text;

GlyphBackColor – glyphs in check-boxes and radio-buttons;

SelectedTextColor, SelectedTextBackgroundColor – selection in edits and memos;

PanelBackgroundColor – panel background;

PanelBorderColor – panel border;

TabColor, HoverTabColor, ActiveTabColor – tabs in tab-sets;

TabCloseButtonColor – close buttons in tab-sets;

TabSeparatorColor – separators in tab-sets;

CheckMarkColor – lines in glyphs in check-boxes and radio-buttons.

8.7 TRVCustomMathEquationManager

This class defines global properties for math expression items [193] in all documents.

Only one object of this class can exist in the application; it is returned by RVMathEquationManager [1054] function.

Unit: RVCustomMathWrapper.

Syntax

```
TRVCustomMathEquationManager = class;
```

(introduced in version 23)

Hierarchy

TObject

Properties

The properties below are used by the commercial version of Adit Math Engine:

- SaveMathMLInHTML [1021] enables saving math items as equation objects (MathML format) in HTML files.
- SaveOMMLInDocX [1021] enables saving math items as equation objects (OMML format) in DocX files.

8.7.1 Properties

In TRVCustomMathEquationManager

SaveMathMLInHTML [1021]
SaveOMMLInDocX [1021]

8.7.1.1 TRVCustomMathEquationManager.SaveMathMLInHTML

Instructs saving math expressions [193] in HTML files in MathML format.

```
property SaveMathMLInHTML: Boolean;
```

If *True*, equations are saved in HTML as equations in MathML (***Mathematical Markup Language***) format.

If *False*, they are saved as images.

This property is used when a commercial version of Adit Math Engine is used. The free version does not support MathML and always saves equations as images.

Default value

True

8.7.1.2 TRVCustomMathEquationManager.SaveOMMLInDocX

Instructs saving math expressions [193] in DocX files in OMML format.

```
property SaveOMMLInDocX: Boolean;
```

If *True*, equations are saved in DocX as equations in OMML (Office Math Markup Language) format.

If *False*, they are saved as images.

This property is used when a commercial version of Adit Math Engine is used. The free version does not support OMML and always saves equations as images.

Default value

True

8.8 TRVBackgroundProperties

The class defines the position, size, and repetition (tiling) of the background image.

Unit [VCL/FMX]: RVBack / fmxRVBack.

Syntax

```
TRVBackgroundProperties = class (TPersistent);
```

(introduced in version 24)

Hierarchy

TObject
TPersistent

Properties

By default, a background image is not shown. Assign Visible [1025] = *True* to enable it.

The position and size of the background image are set separately for horizontal and vertical directions, but there are some dependencies. In particular, the values *rvbszContain* and *rvbszCover* are applied to HorizontalSizeKind [1024] and VerticalSizeKind [1025] simultaneously, while *rvbszAuto* sets one dimension proportionally to the other.

When the size/offset is specified as a percentage, it refers to the percentage of the whole background area, not of the background image itself.

When the image is repeated, the position properties refers to a single instance of the background image; the others are generated by repeating it.

However, the *rvbrpSpace* mode of HorizontalRepeat[1024]/VerticalRepeat[1025] is an exception: in this mode, the image is always repeated from one side to the other (with added spacing between images), and all positioning properties of the image are ignored.

Size properties:

- HorizontalSizeKind[1024] and ImageWidth[1024]
- VerticalSizeKind[1025] and _ImageHeight[1025]

Positioning properties:

- HorizontalAlignment[1023], HorizontalOffset and HorizontalOffsetKind[1023]
- VerticalAlignment[1024], VerticalOffset and VerticalOffsetKind[1025]

Other properties:

- Attachment[1022]
- HorizontalRepeat[1024]
- VerticalRepeat[1025]
- Visible[1025]

8.8.1 Properties

In TRVBackgroundProperties

Attachment[1022]
HorizontalAlignment[1023]
HorizontalOffset[1023]
HorizontalOffsetKind[1023]
HorizontalRepeat[1024]
HorizontalSizeKind[1024]
ImageWidth[1024]
ImageHeight[1025]
VerticalAlignment[1024]
VerticalOffset[1025]
VerticalOffsetKind[1025]
VerticalRepeat[1025]
VerticalSizeKind[1025]
Visible[1025]

8.8.1.1 TRVBackgroundProperties.Attachment

Defines which area the background image fills.

```
type // defined in RVStyle unit
  TRVBackgroundAttachment = (
    rvbatContent, rvbatOwner
```

```
  );

property Attachment:
  TRVBackgroundAttachment;
```

Value	Meaning
rvbatContent	The image fills the content area of the owner object. If the content scrolls, the background scrolls along with it.
rvbatOwner	The image fills the entire area of the owner object itself and does not scroll with its content.

This property is important when applied to the TRichView component itself. For tables, table cells, or printed page backgrounds, this property can be ignored. In those cases, the object's area and its content area coincide, so there is no practical difference between *rvbatContent* and *rvbatOwner*.

Default value

rvbatContent

8.8.1.2 TRVBackgroundProperties.HorizontalAlignment

Horizontal alignment of the background image within the background rectangle.

```
type // defined in RVStyle unit
  TRVBackgroundHorizontalAlignment = (
    rvbhaLeft, rvbhaCenter, rvbhaRight
  );

property HorizontalAlignment:
  TRVBackgroundHorizontalAlignment;
```

This property is ignored if HorizontalRepeat[1024] = *rvbrpSpace*.

Default value

rvbhaLeft

8.8.1.3 TRVBackgroundProperties.HorizontalOffset, HorizontalOffsetKind

Horizontal offset of the background image relative to the position defined by HorizontalAlignment[1023].

```
property HorizontalOffsetKind: TRVDualSignedLengthKind[1066];
property HorizontalOffset: TRVDualSignedLength[1066];
```

These properties are ignored if HorizontalRepeat[1024] = *rvbrpSpace*.

The offset direction depends on HorizontalAlignment[1023]:

- left: positive values shift the image to the right, negative values shift it to the left.
- right: positive values shift the image to the left, negative values shift it to the right.
- center: offset is not applied, these properties are ignored.

When the offset is specified as a percentage, it refers to the percentage of the whole background area, not of the background image itself.

Default value

rvdslkAbsolute, 0

8.8.1.4 TRVBackgroundProperties.HorizontalRepeat

Specifies whether the background image is repeated horizontally, and how it is repeated.

```
property HorizontalRepeat: TRVBackgroundRepeat[1057];
```

Default value

rvbrpRepeat

8.8.1.5 TRVBackgroundProperties.HorizontalSizeKind, ImageWidth

The properties specify the background image width.

```
property HorizontalSizeKind: TRVBackgroundSize[1058];
property ImageWidth: TRVDualLength[1066];
```

ImageWidth is used only if **HorizontalSizeKind** = *rvbszValue*. Negative values represent percentages. Positive values represent units defined in TRVStyle[739].Units[764] (pixels, twips, EMU).

When the size is specified as a percentage, it refers to the percentage of the whole background area, not of the background image itself.

The following values of **HorizontalSizeKind** are special: *rvbszContain* and *rvbszCover*. If you assign one of them to the **HorizontalSizeKind**, it is automatically applied to VerticalSizeKind[1024] as well; and vice versa.

Default value

rvbszAuto, -100000 (100%)

8.8.1.6 TRVBackgroundProperties.VerticalAlignment

Vertical alignment of the background image within the background rectangle.

```
type // defined in RVStyle unit
  TRVBackgroundVerticalAlignment = (
    rvbvaTop, rvbvaCenter, rvbvaBottom
  );

property VerticalAlignment:
  TRVBackgroundVerticalAlignment;
```

This property is ignored if VerticalAlignment[1024] = *rvbrpSpace*.

Default value

rvbvaTop

8.8.1.7 TRVBackgroundProperties.VerticalOffset, VerticalOffsetKind

Vertical offset of the background image relative to the position defined by VerticalAlignment[1024].

```
property HorizontalOffsetKind: TRVDualSignedLengthKind[1066];
property HorizontalOffset: TRVDualSignedLength[1066];
```

These properties are ignored if VerticalAlignment[1025] = *rvbrpSpace*.

The offset direction depends on VerticalAlignment[1024]:

- top: positive values shift the image down, negative values shift it up.
- bottom: positive values shift the image up, negative values shift it down.
- center: offset is not applied, these properties are ignored.

When the offset is specified as a percentage, it refers to the percentage of the whole background area, not of the background image itself.

Default value

rvdslkAbsolute, 0

8.8.1.8 TRVBackgroundProperties.VerticalRepeat

Specifies whether the background image is repeated vertically, and how it is repeated.

```
property VerticalRepeat: TRVBackgroundRepeat[1057];
```

Default value

rvbrpRepeat

8.8.1.9 TRVBackgroundProperties.VerticalSizeKind, ImageHeight

The properties specify the background image height.

```
property VerticalSizeKind: TRVBackgroundSize[1058];
property ImageHeight: TRVDualLength[1066];
```

ImageHeight is used only if **VerticalSizeKind** = *rvbszValue*. Negative values represent percentages. Positive values represent units defined in TRVStyle[739].Units[764] (pixels, twips, EMU).

When the size is specified as a percentage, it refers to the percentage of the whole background area, not of the background image itself.

The following values of **VerticalSizeKind** are special: *rvbszContain* and *rvbszCover*. If you assign one of them to the **VerticalSizeKind**, it is automatically applied to HorizontalSizeKind[1024] as well; and vice versa.

Default value

rvbszAuto, -100000 (100%)

8.8.1.10 TRVBackgroundProperties.Visible

Shows/hides a background image.

```
property Visible: Boolean;
```

Default value

False

8.9 TRVDefaultLoadProperties

This class defines default properties for loading HTML, Markdown, RTF, DocX documents in TRichView [216].

This is a type of global variable RVDefaultLoadProperties [1119].

Unit [VCL/FMX]: RVDefReadProps / fmxRVDefReadProps.

Syntax

```
TRVDefaultLoadProperties = class;
```

(introduced in version 20)

Hierarchy

TObject

8.9.1 Properties

In TRVDefaultLoadProperties

DefaultBulletFontName [1026]
DefaultFontName [1026]
DefaultFontSizeDouble [1027]
DefaultFontSizesDouble [1027]
DefaultMonoSpacedFontName [1028]
DefaultProperties [1028]
DefaultUnicodeSymbolFontName [1028]
GenericFontNames [1028]
ListMarkerIndentPix [1031]

8.9.1.1 TRVDefaultLoadProperties.DefaultBulletFontName

Default font for bullets (paragraph list markers)

```
property DefaultBulletFontName: TFontName;
```

This font is used by default for paragraph bullets loaded from:

- HTML
- Markdown
- RTF and DocX, if RTFReadProperties [253].ConvertSymbolFonts [461] = *True*.

This property provides access to GenericFontNames [1028][*rvgffRVBullets*].

8.9.1.2 TRVDefaultLoadProperties.DefaultFontName

Default HTML font.

```
property DefaultFontName: TFontName;
```

This property provides access to GenericFontNames [1028][*rvgffRVDefault*].

8.9.1.3 TRVDefaultLoadProperties.DefaultFontSizeDouble

Defines fonts size for HTML import, in half-points.

```
property DefaultFontSizeDouble: Integer;
```

This property provides access to DefaultFontSizesDouble[1027][*rvfstDefault*]

8.9.1.4 TRVDefaultLoadProperties.DefaultFontSizesDouble

Defines fonts sizes for HTML import.

```
type
  TRVFontSizeType = (
    rvfstDefault,
    rvfstXXSmall, rvfstXSmall, rvfstSmall,
    rvfstMedium,
    rvfstLarge, rvfstXLarge, rvfstXXLarge,
    rvfstXXXLarge);
```

```
property DefaultFontSizesDouble[
  Index: TRVFontSizeType]: Integer
```

Values are measured in half-points. For example, value 24 means 24/2 = 12 pt.

The following fonts are used when importing HTML files.

Value	HTML <font size>	CSS font-size	Default Value
rvfstDefault	Used if not defined		24
rvfstXXSmall	1	xx-small	16
rvfstXSmall	–	x-small	18
rvfstSmall	2	small	20
rvfstMedium	3	medium	24
rvfstLarge	4	large	28
rvfstXLarge	5	x-large	36
rvfstXXLarge	6	xx-large	48
rvfstXXXLarge	7	xxx-large	56

It's highly recommend to set **DefaultFontSizesDouble**[*rvfstDefault*] equal to **DefaultFontSizesDouble**[*rvfstMedium*].

8.9.1.5 TRVDefaultLoadProperties.DefaultMonoSpacedFontName

Default monospaced HTML and Markdown font.

```
property DefaultMonoSpacedFontName: TFontName;
```

This property provides access to GenericFontNames[1028] [*rvgffUIMonospace*].

8.9.1.6 TRVDefaultLoadProperties.DefaultProperties

Default text attributes for Markdown import.

```
type
  TRVFormattingType = (
    rvftH1, rvftH2, rvftH3, rvftH4, rvftH5,
    rvftH6, rvftHyperlink);

 TRVFormattingDefaultItem = record
    FontSizeDouble: Integer;
    FontName: TFontName;
    Color: TRVColor[1063];
    Style: TFontStyles;
  end;

property DefaultProperties[
  Index: TRVFormattingType]: TRVFormattingDefaultItem;
```

if StyleTemplates are not used[263], or there are no necessary style templates, this property defines attributes of headings and hyperlinks.

8.9.1.7 TRVDefaultLoadProperties.DefaultUnicodeSymbolFontName

A font name that is used instead of symbol fonts when importing RTF and DocX files.

```
property DefaultUnicodeSymbolFontName: TFontName;
```

This font is used if RTFReadProperties[253].ConvertSymbolFonts[461] = *True*.

This property provides access to GenericFontNames[1028] [*rvgffRVSymbols*].

8.9.1.8 TRVDefaultLoadProperties.GenericFontNames

Default font names.

```
type
  TRVGenericFontFamily = (
   rvgffRVDefault, rvgffSerif, rvgffSansSerif,
   rvgffMonospace, rvgffCursive, rvgffFantasy,
   rvgffSystemUI, rvgffUISerif, rvgffUISansSerif,
   rvgffUIMonospace, rvgffUIRrounded, rvgffEmoji,
   rvgffMath, rvgffFangsong, rvgffRVSymbols,
   rvgffRVBullets);

property GenericFontNames[
```

```
  Index: TRVGenericFontFamily]: TFontName;
```

HTML import

The following fonts are used when importing HTML files.

Value	Used
rvgffRVDefault	If font is not defined
rvgffSerif	For CSS “serif” font family
rvgffSansSerif	For CSS “sans-serif” font family
rvgffMonospace	For CSS “monospace” font family (and for tags like <code>)
rvgffCursive	For CSS “cursive” font family
rvgffFantasy	For CSS “fantasy” font family
rvgffSystemUI	For CSS “system-ui” font family
rvgffUISansSerif	For CSS “ui-sans-serif” font family
rvgffUISerif	For CSS “ui-serif” font family
rvgffUIMonospace	For CSS “ui-monospace” font family
rvgffUIRrounded	For CSS “ui-rounded” font family
rvgffEmoji	For CSS “emoji” font family
rvgffMath	For CSS “math” font family
rvgffFangsong	For CSS “fangsong” font family
rvgffRVBullets	For bullets (paragraph list markers)

Markdown

The following fonts are used when importing Markdown files.

Value	Used
rvgffMonospace	For code fragments, if StyleTemplates are not used [263], or there is no 'HTML Code' style template.
rvgffRVBullets	For bullets (paragraph list markers)

RTF and DocX

The following fonts are used when importing RTF and DocX files.

Value	Used
rvgffRVSymbols	Instead of symbol fonts, if ConvertSymbolFonts [461] = *True*
rvgffRVBullets	For bullets (paragraph list markers) instead of symbol fonts, if ConvertSymbolFonts [461] = *True*

Default values

Default values depend on the platform.

Value	Default Value				
	Windows	**macOS**	**iOS**	**Linux**	**Android**
rvgffRVDefault	'Tahoma'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
rvgffSerif	'Times New Roman'	'Times'	'Times New Roman'	'Liberation Serif'	'serif'
rvgffSansSerif	'Arial'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
rvgffMonospace	'Courier New'	'Courier New'	'Courier New'	'DejaVu Sans Mono'	'monospace'
rvgffCursive	'Comic Sans MS'	'Apple Chancery'	'Snell Roundhand'	'Z003'	'normal'
rvgffFantasy	'Impact'	'Papyrus'	'Papyrus'	'Impact'	'normal'
rvgffSystemUI	'Segoe UI'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
rvgffUISansSerif	Same as for *rvgffSansSerif*				
rvgffUISerif	Same as for *rvgffSerif*				
rvgffUIMonospace	Same as for *rvgffMonospace*				
rvgffUIRrounded	Same as for *rvgffSansSerif*				
rvgffEmoji					
rvgffMath					
rvgffFangsong	'FangSong'	'FangSong'	'FangSong'	'FangSong'	'FangSong'
rvgffRVSymbols	'Segoe UI Symbol'	'Apple Symbols'	'Apple Symbols'	'Liberation Sans'	'normal'

rvgffRVBullets	Same as for *rvgffRVDefault*

8.9.1.9 TRVDefaultLoadProperties.ListMarkerIndentPix

A list marker indent for bullets and numbering imported from HTML and Markdown.

```
property ListMarkerIndentPix: TRVPixel96Length[1085];
```

This is the default distance between the left side of a list marker and the left side of its paragraph text.

Default value:

24

8.10 TRVDeleteUnusedStylesData

This class allows to delete unused text[764], paragraph[757] and list[755] styles in multiple documents (to delete styles that do not present in at least one of the documents)

Unit [VCL/FMX]: RVItem / fmxRVItem.

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[129] (but stored in file or in the Registry). In this case, several documents share the same collections of styles. TCustomRichView.DeleteUnusedStyles[293] cannot be used in this case, MarkStylesInUse[339] and DeleteMarkedStyles[292] 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[339](data);
  RichView2.MarkStylesInUse[339](data);
  RichView1.DeleteMarkedStyles[292](data);
  RichView2.DeleteMarkedStyles[292](data);
  data.Free;
```

Example 2

How to remove unused styles in several documents.

```
uses CRVData, RVItem;
```

```
data := TRVDeleteUnusedStylesData.Create(True, True, True);
RichView1.LoadRVF(336)('Doc1.rvf');
RichView1.MarkStylesInUse(339)(data);
RichView1.LoadRVF(336)('Doc2.rvf');
RichView1.MarkStylesInUse(339)(data);
...
RichViewDoNotCheckRVFStyleRefs(1112) := True;
RichView1.LoadRVF(336)('Doc1.rvf');
RichView1.DeleteMarkedStyles(292)(data);
RichView1.SaveRVF(351)('Doc1.rvf', False);
RichView1.LoadRVF(336)('Doc2.rvf');
RichView1.DeleteMarkedStyles(292)(data);
RichView1.SaveRVF(351)('Doc2.rvf', False);
...
RichViewDoNotCheckRVFStyleRefs(1112) := False;
data.Free;
```

8.11 TRVFont

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVFont = class (TPersistent);
```

Font, analog of TFont.

It has properties: Color, Size, Charset, Name, Style, Height.

Unlike TFont, TRVFont stores the Size property instead of Height.

8.12 TRVGraphic

A class for objects representing images (bitmaps, PNG, etc.)

Unit [VCL/FMX]: RVType / fmxRVType;

Syntax [VCL]:

```
TRVGraphic = TGraphic;
```

Syntax [FMX]:

```
TRVGraphic = TRVGraphicFM;
TRVGraphicFM = class (TPersistent);
// TRVGraphicFM is defined in fmxRVGraphicsFM unit
```

Description [VCL]

In VCL (and LCL), this is the abstract base class type for objects such as icons, bitmaps, and metafiles that can store and display visual images. Classes inherited from TGraphic are used (such as TBitmap, TPngImage, etc.)

Description [FireMonkey]

In FireMonkey, classes of graphic types are inherited from the abstract TRVGraphicFM class (also accessible as TRVGraphic).

Two classes are inherited from TRVGraphicFM:

- TRVRasterGraphicFM – wrapper for FireMonkey's TBitmap (fmxRVGraphicsFM unit), also accessible as TRVBitmap
- TRVSvgImageSkiaFM – wrapper for Skia4Delphi's TSkSvgBrush (fmxRVSvgSkiaFM unit), represents an SVG image

TRVRasterGraphicFM has two properties:

- Bitmap: TBitmap
- GraphicType: TRVGraphicType[1077]

GraphicType specifies a format for saving this image.

You can use CreateFromBitmap constructor to create a TRVRasterGraphicFM object for the specified TBitmap object.

```
constructor CreateFromBitmap(ABitmap: TBitmap; SharedBitmap: Boolean);
```

If SharedBitmap = False, ABitmap will be owned by the created object (and will be freed together with it).

If SharedBitmap = True, a copy of ABitmap will be created and used by the created object.

TRVSvgImageSkiaFM has Svg: TSkSvgBrush property

Note: you can use RVGraphicHandler[1125].GetFileDialogFilter[1033] to get a filter for opening and saving dialogs. It returns a filter string containing not only raster formats (registered for TBitmap), but also additional formats (such as SVG), if they are available.

8.13 TRVGraphicClass

A class type for graphic object.

Unit [VCL/FMX]: RVClasses / fmxRVClasses;

Syntax [VCL]:

```
TRVGraphicClass = TGraphicClass;
```

Syntax [FMX]:

```
TRVGraphicClass = class of TRVGraphicFM[1032];
```

8.14 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 [VCL/FMX]: RVGrHandler / fmxRVGrHandler

Syntax

```
TRVGraphicHandler = class;
```

(introduced in version 15)

Using

A single variable of this class is created: RVGraphicHandler[1125]. 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:

```
procedure SetDefaultGraphicClass(GraphicType: TRVGraphicType[1077];
  GraphicClass: TRVGraphicClass[1033]);
procedure RegisterPngGraphic(GraphicClass: TRVGraphicClass[1033]);
procedure RegisterJpegGraphic(GraphicClass: TRVGraphicClass[1033]);
procedure RegisterHTMLGraphicFormat(GraphicClass: TRVGraphicClass[1033]);
function CreateGraphicByType(GraphicType: TRVGraphicType[1077]): TRVGraphic[1032];
function LoadFromFile(const FileName: TRVUnicodeString[1099]): TRVGraphic[1032];
function LoadFromStream(Stream: TStream): TRVGraphic[1032];
function GetDataGraphicType(const Buffer; Size: Integer): TRVGraphicType[1077];

// VCL and LCL
function CreateGraphic(GraphicClass: TRVGraphicClass[1033]): TRVGraphic[1032];
// FireMonkey
function GetFileDialogFilter: String;
```

[FireMonkey] Normally, to get a filter string for a file dialog to open image files, developers use TBitmapCodecManager.GetFilterString. However, this filter string does not include non-bitmap formats supported by TRichView (such as SVG supported using Skia4Delphi). So, you can use **GetFileDialogFilter** method to get a filter string that includes both bitmap and non-bitmap formats.

Graphic classes

In VCL and Lazarus, different graphic classes represent images. All of them are inherited from TGraphic.

In FireMonkey, graphic classes are inherited fromTRVGraphicFM[1032] (also available as TRVGraphic). There are two classes that can be used:

- TRVRasterGraphicFM (also available as TRVBitmap): wrapper for FireMonkey TBitmap, supports many raster image formats
- TRVSvgImageSkiaFM: wrapper for TSkSvgBrush from Skia4Delphi, supports SVG images.

Choosing default graphic classes [VCL and LCL]

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 DocX files. Optionally, bitmaps can be saved as PNG in RTF too, see TCustomRichView.RTFOptions[252] 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). In Lazarus [160], TPortableNetworkGraphic is registered automatically.

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 (for Delphi), undefined for Lazarus [160];
- jpeg: TJPEGImage;
- png: TPngImage (for Delphi 2009+), TPortableNetworkGraphic (for Lazarus [160]), 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);
- anymap: TPortableAnyMapGraphic (for Lazarus [160]), undefined otherwise.
- svg: TSkSvgGraphic (requires Skia4Delphi); include RVSkia unit in your project.

HTML export [VCL and LCL]

When saving image to HTML or Markdown, TRichView converts all non-HTML image formats as to Jpegs.

Images of graphic classes registered for the following formats are not converted: rvgtJPEG, rvgtPNG, rvgtGIF, rvgtSVG, rvgtWebP.

RegisterHTMLGraphicFormat allows to define additional HTML graphic formats. Call this method when the application starts (for example, in the initialization section of the main form's unit). Images of these formats are saved in HTML [132] as they are, without conversion to Jpegs. 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 TRVGraphic [1032] TRichView tries to recognize the proper graphic class by content itself. If failed, it tries to use framework methods to recognize an load an image (in VCL/LCL: by file extension; in FireMonkey: by content). If failed, it 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, it returns *nil*.

See also

how to use third-party graphic classes in TRichView[1136]

8.15 TRVIntegerList

List of integer values, used internally by TRichView classes.

Unit [VCL/FMX] RVClasses / fmxRVClasses;

```
type
  TRVIntegerList = class(TList)
```

Main properties:

Items[Index: Integer]: Integer.

Count: Integer

8.16 TRVItemFactory

A singleton class for creating items in a document.

Unit [VCL/FMX] RVItems / fmxRVItems;

```
type
  TRVItemFactory = class;
```

Main properties:

- TextItemClass: TRVTextItemInfoClass;
- GraphicItemClass: TRVGraphicItemInfoClass;
- HotGraphicItemClass: TRVGraphicItemInfoClass;
- ControlItemClass: TRVControlItemInfoClass;
- TabItemClass: TRVTabItemInfoClass;
- BulletItemClass: TRVBulletItemInfoClass;
- HotspotItemClass: TRVHotspotItemInfoClass;
- BreakItemClass: TRVBreakItemInfoClass;
- MarkerItemClass: TRVSimpleRectItemInfoClass;

where

```
  TCustomRVItemInfoClass =
    class of TCustomRVItemInfo[899];
  TRVTextItemInfoClass =
    class of TRVTextItemInfo;
  TRVSimpleRectItemInfoClass =
    class of TRVSimpleRectItemInfo;
  TRVTabItemInfoClass =
    class of TRVTabItemInfo;
  TRVControlItemInfoClass =
```

```
    class of TRVControlItemInfo;
  TRVGraphicItemInfoClass =
    class of TRVGraphicItemInfo;
  TRVBulletItemInfoClass =
    class of TRVBulletItemInfo;
  TRVHotspotItemInfoClass =
    class of TRVHotspotItemInfo;
  TRVBreakItemInfoClass =
    class of TRVBreakItemInfo;
```

The object of this class is returned by RVItemFactory[1054] function.

8.17 TRVMathDocObject

This is a class of a document object defining default properties for items-mathematical expressions[193].

Unit RVMathItem;

Syntax

```
TRVMathDocObject = class (TRVDocObject[235]);
```

(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[976], FontSizeDouble[976], TextColor[977] properties.

See the explanations about mathematical fonts in the help topics about TRVMathItemInfo.FontName[976]

How to use

It's not recommended to use this object directly. Use RVGetMathProperties and RVSetMathProperties[1048] procedures instead.

To save and load these properties in RVF[129], include *rvfoSaveDocObjects* and *rvfoLoadDocObjects* in RVFOptions[254].

8.18 TRVPicture

A class for objects that contains TRVGraphic[1032] objects inside.

Unit [VCL/FMX]: RVType / fmxRVType;

Syntax [VCL]:

```
TRVPicture = TPicture;
```

Syntax [FMX]:

```
TRVPicture = TRVPictureFM;
TRVPictureFM = class (TPersistent);
// TRVPictureFM is defined in fmxRVGraphicsFM unit
```

Description

TRVPicture is a TRVGraphic[1032] container.

In FireMonkey, TRVPictureFM is implemented as an analog of TPicture, to allow using the same graphic properties.

See also:

- TRVStyle[739].InvalidPicture[752]
- TRVListLevel[859].Picture[868]

8.19 TRVRect

This is a class similar to TRect record.

Unit [VCL/FMX] RVStyle / fmxRVStyle;

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[1094] type.

8.20 TRVReportHelperWithHeaderFooters

This class contains TRVReportHelper[721] components representing a main document and all six possible headers and footers.

Unit [VCL/FMX]: RVWithHeaderFooter / fmxRVWithHeaderFooter.

Syntax

```
TRVReportHelperWithHeaderFooters = class;
```

(introduced in version 17)

Hierarchy

TObject

Descriptions

An object of this class contains:

- TRVReportHelper[721] component used for the main document (it is accessible as MainDoc[1039])

- TRVReportHelper components used for all six headers and footers (they are accessible as GetHeader[1040] and GetFooter[1040])
- TRVStyle[739] components for each TRVReportHelper.

See details in the help topic about the constructor[1040].

Use AssignToRVPrint[1040] to initialize TRVPrint[696] 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***)

8.20.1 Properties

In TRVReportHelperWithHeaderFooters

- MainDoc[1039]
- UseStyleTemplates[1039]

8.20.1.1 TRVReportHelperWithHeaderFooters.MainDoc

Returns TRVReportHelper component used to store the main document.

```
property MainDoc: TRVReportHelper[721];
```

See also

- GetHeader[1040]
- GetFooter[1040]

8.20.1.2 TRVReportHelperWithHeaderFooters.UseStyleTemplates

Allows or disallows using MainDoc[1039].RichView[724].Style[260].StyleTemplates[762].

```
property UseStyleTemplates: Boolean;
```

This property provides access to UseStyleTemplates[263] property of all RichViews of all TRVReportHelpers stored in the component.

Default value

False

8.20.2 Methods

In TRVReportHelperWithHeaderFooters

- AssignToRVPrint[1040]
- Create[1040]
- GetFooter[1040]

GetHeader[1040]
Reset[1041]

8.20.2.1 TRVReportHelperWithHeaderFooters.AssignToRVPrint

Prepares the document stored in this component to printing by **RVPrint**.

```
procedure AssignToRVPrint(RVPrint: TRVPrint[696]);
```

This procedure prepares **RVPrint** by calling its AssignSource[709], SetHeader[716], SetFooter[715], AssignDocParameters[708] methods.

This procedure does not format RVPrint (it does not call its FormatPages[711]).

This procedure does not assign page size (only margins and header&footer positions)

8.20.2.2 TRVReportHelperWithHeaderFooters.Create

Creates an instance of TRVReportHelperWithHeaderFooters[1038] component.

```
constructor Create;
```

The constructor:

- creates TRVReportHelper[721] component that will be used for the main document (it is accessible as MainDoc[1039])
- creates TRVReportHelper components that will be used for all six headers and footers (they are accessible as GetHeader[1040] and GetFooter[1040]), assigns them as headers/footers to the main document;
- creates TRVStyle[739] 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[756] properties to MainDoc[1039].RichView[724].Style[260] (so, if you decide to use style templates[1039], they all will use MainDoc[1039].RichView[724].Style[260].StyleTemplates[762].

Default value

False

8.20.2.3 TRVReportHelperWithHeaderFooters.GetFooter

Returns TRVReportHelper component representing a footer of the specified type.

```
function GetFooter(FooterType: TRVHFType[1078]): TRVReportHelper[721];
```

See also

- GetHeader[1040]
- MainDoc[1039]

8.20.2.4 TRVReportHelperWithHeaderFooters.GetHeader

Returns TRVReportHelper component representing a header of the specified type.

```
function GetHeader(FooterType: TRVHFType[1078]): TRVReportHelper[721];
```

See also

- GetFooter[1040]
- MainDoc[1039]

8.20.2.5 TRVReportHelperWithHeaderFooters.Reset

Completely clears all documents stored in this component.

```
procedure Reset;
```

This procedure clears [286] all documents (the main document, headers, footers), deletes all text [764], paragraph [757] and list [755] styles from all TRVStyles, adds one text and one text and paragraph style having default properties.

If UseStyleTemplates [1039] = True, it also clears StyleTemplates [762] for the main document, adds a single normal style template [799], and assign it to paragraph styles.

Additionally, it assigns 5 pixels to all margins (LeftMargin [245], RightMargin [251], TopMargin [262], BottomMargin [231]), and resets DocParameters [237] (keeping their Units [428]).

8.21 TRVStringList

A string list capable to store Unicode strings in all versions of Delphi.

Unit [VCL/FMX]: RVClasses / fmxRVClasses;

Syntax:

```
TRVStringList = class (TStringList)
```

Description

For Unicode versions of Delphi, TRVStringList is a TStringList with some additional methods.

For old versions of Delphi and Lazarus, it stores strings as UTF-8 inside, but methods work with TRVUnicodeString parameters. There are some unsupported features in this mode: Sorted property and andling duplicates. However, the following methods work and perform correct comparisons: Sort, FindUnicodeString, IndexOfUnicodeString, IndexOfUnicodeName.

Most important methods:

```
function AddUnicodeString(
  const S: TRVUnicodeString[1099]): Integer;
function GetUnicodeString(
  Index: Integer): TRVUnicodeString[1099];
procedure SetUnicodeString(Index: Integer;
  const Value: TRVUnicodeString[1099]);
```

8.22 TRVThumbnailMaker

This class creates smooth scaled copies of images.

Unit [VCL/FMX]: RVThumbMaker / fmxRVThumbMaker;

Syntax

```
TRVThumbnailMaker = class;
```

(introduced in version 15)

Using

A single variable of this class is created: RVThumbnailMaker[1129]. 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

For VCL and LCL:

```
procedure AllowThumbnailsFor(Cls: TGraphicClass);
procedure DisallowThumbnailsFor(Cls: TGraphicClass);
```

For all frameworks:

```
function IsAllowedFor(Gr: TRVGraphic[1032]): Boolean;
function MakeThumbnail(Gr: TRVGraphic[1032]; Width, Height: Integer): TBitmap;
```

AllowThumbnailsFor allows making scaled images for the specified graphic class.

Delphi: 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). For old versions of Delphi (Delphi 2007 and older), thumbnails are not created for transparent images.

Lazarus: By default, scaling is allowed for all images of classes inherited from TFPImageBitmap (that includes all Lazarus raster graphic formats).

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[125]

8.23 TRVUnitsRect

This class contains 4 properties of TRVLength[1081] type: Left, Top, Right, Bottom

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
TRVFloatRect = class (TPersistent)
```

(Introduced in version 15)

This class is used for Margins [705] property of TRVPrint [696].

9 Procedures and functions

Functions returning singleton objects

RVItemFactory [1054] creates items in documents [163].

RVMathEquationManager [1054] [VCL] returns the object containing global properties used by equation objects [193].

Other TRichView procedures and functions

DrawControl [1043] (and other functions) draws controls on the specified canvas [VCL and LCL]

GetRVESearchOptions [1049] converts options of a find/search dialog to TRichViewEdit search options [VCL and LCL]

GetRVSearchOptions [1049] converts options of a find/search dialog to TRichView search options [VCL and LCL]

RV_GetPrinterDC [1050] creates a DC compatible with the active printer [VCL and LCL]

RVGetFirstEndnote [1052] (and other functions) enumerates notes in the document

RVGetNoteTextStyleNo [1052] returns a text file for footnotes or endnotes

RVIsURL, RVIsEmail [1053] check whether the string contains an URL or e-mail address

RVOpenURL [1054] opens the specified URL in an external browser

Unicode conversion [1054] function convert text to and from Unicode

Functions from RVGetText unit [1045] returns text representation of the document or its fragment

Functions from RVLinear unit [1047] provide 1:1 conversion between the document and a text string

Functions from RVMathItem unit [1048] help to work with common properties of equation objects [193]

Functions for TRVUnits conversion [1044]

Functions for extraction HTML from RTF [1049]

9.1 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 [174], see OnPrintComponent [887] event.

9.2 Functions for TRVUnits conversion

unit [VCL/FMX] RVFuncs / fmxRVFuncs;

```
function RV_UnitsPerInch(Units: TRVUnits[1099];
  Control: TControl): TRVLength[1081]; overload;
function RV_UnitsPerInch(Units: TRVUnits[1099];
  PixelsPerInch: Integer): TRVLength[1081]; overload;

function RV_UnitsToTwips(Value: TRVLength[1081]; Units: TRVUnits[1099];
  Control: TControl): Integer; overload; inline;
function RV_UnitsToTwips(Value: TRVLength[1081]; Units: TRVUnits[1099];
  PixelsPerInch: Integer): Integer; overload; inline;

function RV_UnitsToEMU(Value: TRVLength[1081]; Units: TRVUnits[1099];
  Control: TControl): Integer; overload; inline;
function RV_UnitsToEMU(Value: TRVLength[1081]; Units: TRVUnits[1099];
  PixelsPerInch: Integer): Integer; overload; inline;

function RV_UnitsToPixels(Value: TRVLength[1081]; Units: TRVUnits[1099];
  Control: TControl): Integer; overload; inline;
function RV_UnitsToPixels(Value: TRVLength[1081]; Units: TRVUnits[1099];
  PixelsPerInch: Integer): TRVCoord[1064]; overload; inline;

function RV_TwipsToUnits(Value: Integer; Units: TRVUnits[1099];
  Control: TControl): TRVLength[1081]; overload; inline;
function RV_TwipsToUnits(Value: Integer; Units: TRVUnits[1099];
  PixelsPerInch: Integer): TRVLength[1081]; overload; inline;

function RV_EMUToUnits(Value: Integer; Units: TRVUnits[1099];
  Control: TControl): TRVLength[1081]; overload; inline;
function RV_EMUToUnits(Value: Integer; Units: TRVUnits[1099];
  PixelsPerInch: Integer): TRVLength[1081]; overload; inline;

function RV_UnitsToUnits(Value: TRVLength[1081];
  OldUnits, NewUnits: TRVUnits[1099];
  Control: TControl): TRVLength[1081]; overload; inline;
function RV_UnitsToUnits(Value: TRVLength[1081];
  OldUnits, NewUnits: TRVUnits[1099];
  PixelsPerInch: Integer): TRVLength[1081]; overload; inline;
```

RV_UnitsPerInch returns the amount of Units in 1 inch.

RV_UnitsToTwips converts Value from Units to twips.

RV_UnitsToEMU converts Value from Units to EMU.

RV_UnitsToPixels converts Value from Units to pixels.

RV_TwipsToUnits converts Value from twips to Units.

RV_EMUToUnits converts Value from EMU to Units.

RV_UnitsToUnits converts Value from OldUnits to NewUnits.

There are two versions of each of these methods. The first of them has **Control** parameter, the second of them has **PixelsPerInch**. These parameters are used when converting to/from pixels.

For methods with **PixelsPerInch** parameter, 1 pixel = 1/**PixelsPerInch** of an inch.

For methods with **Control** parameter, "pixels per inch" value is determined from **Control** using the following rules:

- if value of the global variable RichViewPixelsPerInch[1115] is positive, it is used (**Control** is ignored); otherwise
- if the application supports multiple monitors having different pixel depth (Delphi 10.1+), DPI of the monitor that contains the form that contains **Control** is used; otherwise
- Screen.PixelsPerInch is used (**Control** is ignored)

See also

- Units of measurement in TRichView[144]

9.3 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.

Unit [VCL/FMX]: RVGetText / fmxRVGetText:

```
function GetCurrentLineText(rve: TCustomRichViewEdit[469];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetVisibleText(rv: TCustomRichView[216];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetAllText(rv: TCustomRichView[216];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetRVDataText(RVData: TCustomRVData[1014];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetCurrentParaSectionText(rve: TCustomRichViewEdit[469];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetCurrentParaText(rve: TCustomRichViewEdit[469];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetCurrentChar(rve: TCustomRichViewEdit[469];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
function GetCurrentWord(rve: TCustomRichViewEdit[469];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[1056];
```

Unit [VCL/FMX] RVGetTextW / fmxRVGetTextW:

```
function GetCurrentLineText(rve: TCustomRichViewEdit[469];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetVisibleText(rv: TCustomRichView[216];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetAllText(rv: TCustomRichView[216];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetRVDataText(RVData: TCustomRVData[1014];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetCurrentParaSectionText(rve: TCustomRichViewEdit[469];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetCurrentParaText(rve: TCustomRichViewEdit[469];
  ForANSI: Boolean = False): TRVUnicodeString[1099];
function GetCurrentChar(rve: TCustomRichViewEdit[469]): TRVUnicodeString[1099];
function GetCurrentWord(rve: TCustomRichViewEdit[469]): TRVUnicodeString[1099];
```

(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[1047]

9.4 Functions from RVLinear Unit

Unit [VCL / FMX] RVLinear / fmxRVLinear.

```
const
  RVCharsPerLineBreak: Integer = 2;
  RVNonTextCharacter: TRVUnicodeChar = ' ';
```

```
function RVGetLinearCaretPos(rve: TCustomRichViewEdit[469];
  CharsPerLineBreak: Integer = -1): Integer;
procedure RVSetLinearCaretPos(rve: TCustomRichViewEdit[469];
  LinearPos: Integer; CharsPerLineBreak: Integer = -1);
```

RVGetLinearCaretPos returns the caret position (as a number of characters before the caret). Non-text objects (except for tables[179] and tabulators[168]) are treated as one character, line breaks are treated as CharsPerLineBreak characters.

RVSetLinearCaretPos moves the caret to the specified position.

Value of CharsPerLineBreak parameter	Meaning
2	Use #13#10 to separate lines
1	Use #13 to separate lines
0	Lines are not separated
-1	Use the value of **RVCharsPerLineBreak** global variable (which, in its order, may be 2, 1, or 0)

```
procedure RVGetSelection(rv: TCustomRichView[216];
  out SelStart, SelLength: Integer;
  CharsPerLineBreak: Integer = -1);
procedure RVSetSelection(rv: TCustomRichView[216];
  SelStart, SelLength: Integer;
  CharsPerLineBreak: Integer = -1);
```

RVGetSelection returns the selection position in memo/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: TCustomRichView[216];
  out Selection: TRVSelection;
  CharsPerLineBreak: Integer = -1);
procedure RVSetSelectionEx(rv: TCustomRichView[216];
  const Selection: TRVSelection;
  CharsPerLineBreak: Integer = -1);
```

RVGetSelectionEx returns the selection position in TRVSelection record. **RVSetSelectionEx** sets selection by this record. These functions can handle not only a normal selection, but a multicell selection in tables as well.

```
function RichViewToLinear(rv: TCustomRichView[216];
  CurRVData, RVData: TCustomRVData[1014];
  ItemNo, ItemOffs: Integer; out LinearPos: Integer;
  CharsPerLineBreak: Integer = -1): Boolean;
function LinearToRichView(rv: TCustomRichView[216];
  CurRVData: TCustomRVData[1014];
  var LinearPos: Integer; out RVData: TCustomRVData;
  out ItemNo, ItemOffs: Integer;
  CharsPerLineBreak: Integer = -1): Boolean;
```

RichViewToLinear converts the position specified in TRichView coordinates (RVData – document, ItemNo – index of item in this document, ItemOffs – offset in this item) to a linear coordinate (a number of characters from the beginning of CurRVData before this position). Non-text objects (except for tables[179] and tabulators[168]) are treated as one character, line breaks are treated as CharsPerLineBreak characters. **LinearToRichView** performs an opposite conversion.

```
function RVGetTextRange(rv: TCustomRichView[216];
  RangeStart, RangeLength: Integer;
  CharsPerLineBreak: Integer = -1): TRVUnicodeString[1099];
function RVGetTextLength(rv: TCustomRichView[216];
  CharsPerLineBreak: Integer = -1): Integer;
```

RVGetTextRange returns text from rv, from RangeStart to RangeLength positions. Non-text items (except for the tables[179] and tabulators[168]) are returned as RVNonTextCharacter. As for line breaks, see the table about CharsPerLineBreak above.

The call RVGetTextRange(rv, 0, RVGetTextLength(rv)) returns the whole text.

All these functions are compatible, they calculate position identically (providing that the same value of CharsPerLineBreak is used). A text returned by RVGetTextRange has one-to-one correspondence to TRichView document (unlike text returned by the functions from RVGetText[1045] unit).

9.5 Functions from RVMathItem Unit

```
procedure RVGetMathProperties(rv: TCustomRichView[216];
  var FontName: TFontName; var FontSizeDouble: Integer;
  var TextColor: TColor);
procedure RVSetMathProperties(rv: TCustomRichView[216];
  const FontName: String; FontSizeDouble: Integer;
  TextColor: TColor; EditMode: Boolean);
```

RVGetMathProperties returns default values of properties of mathematical expressions[193]. If **rv**.DocObjects[235] has an item of TRVMathDocObject[1037] 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[235] collection. If such an item does not exist, the procedure creates it. If **EditMode** = *True* and (**rv** is TCustomRichViewEdit[469]), this assignment is performed as an editing operation (undoable).

```
procedure RVApplyMathProperties(rv: TCustomRichView[216]);
```

RVApplyMathProperties updates mathematical expressions after changing properties of TRVMathDocObject[1037] item. You must call this function if you assigned properties of this item directly. It is not needed if you use **RVSetMathProperties.**

9.6 GetRVESearchOptions

Converts options of TFindDialog/TReplaceDialog to TRichViewEdit[469] search options.

Unit RVMisc.

Syntax

```
function GetRVESearchOptions(fo: TFindOptions): TRVESearchOptions[1066];
```

This function always includes *rvseoMultiItem* and *rvseoSmartStart* in the result.

See also

- TRichViewEdit.SearchText[553].

9.7 GetRVSearchOptions

Converts options of TFindDialog to TRichView[216] search options

Unit RVMisc.

Syntax

```
function GetRVSearchOptions(fo: TFindOptions): TRVSearchOptions[1091];
```

This function always includes *rvsroMultiItem* and *rvsroSmartStart* in the result.

See also

- TRichView.SearchText[354].

9.8 HTML from RTF de-encapsulation

unit [VCL/FMX] RVRTFDeEncapsulation / fmxRVRTFDeEncapsulation;

```
type
  TRVAttachmentPositionEvent = procedure (Pos: Integer;
    var TextToInsert: TRVUnicodeString[1099]) of object;

  TRVRTFEncapsulationType = (rvrtfetNone, rvrtfetHTML,
    rvrtfetText, rvrtfetError);

function GetRTFEncapsulationType(
  RTFStream: TStream): TRVRTFEncapsulationType;
function ExtractEncapsulatedHTMLFromRTF(RTFStream, HTMLStream: TStream;
  OnAttachmentPosition: TRVAttachmentPositionEvent = nil): Boolean;
```

HTML-in-RTF encapsulation is performed by Microsoft Outlook/Exchange. A result is an RTF file created from HTML. It looks close to the original HTML file (i.e. it contains RTF code that mimics the original HTML document), but also contains data allowing to extract the original HTML document unchanged.

GetRTFEncapsulationType checks if the specified stream contains HTML or a plain text encapsulated in RTF. Possible result values:

- *rvrtfetNone* –- RTF without encapsulation
- *rvrtfetHTML*– RTF containing encapsulated HTML
- *rvrtfetText* – RTF containing encapsulated plain text
- *rvrtfetError* – not RTF

ExtractEncapsulatedHTMLFromRTF performs extraction of encapsulated HTML or plain text from RTF (from RTFStream to HTMLStream).

Also, if the callback function OnAttachmentPosition is provided, it calls it when it finds an attachment position (you can store the position of HTMLStream where this attachment is located, and/or insert some text in resulting HTML/text in this position).

See also

- ***Additional information about HTML-in-RTF encapsulation***

9.9 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 “standard” pixels. Document area is paper size minus margins.

For conversion, it uses screen resolution (RichViewPixelsPerInch[1115] is ignored).

It does not take headers[716] and footers[715] into account.

```
procedure GetPageSize(RVPrint: TRVPrint[696]; ARVStyle: TRVStyle[739];
  out Width, Height: TRVPixelLength[1085]);
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(1044)(RVPrint.Margins(705).Left,
    RVPrint.Units(707), lpx)- phoX;
  TM := RV_UnitsToPixels(RVPrint.Margins.Top,
    RVPrint.Units, lpy)- phoY;
  RM := RV_UnitsToPixels(RVPrint.Margins.Right,
    RVPrint.Units, lpx) - (phW - (phoX + Width));
  BM := RV_UnitsToPixels(RVPrint.Margins.Bottom,
    RVPrint.Units, lpy) - (phH - (phoY + Height));

  if RVPrint.FixMarginsMode(703)=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);

  Width  := MulDiv(Width,  ARVStyle.UnitsPixelsPerInch(765), lpx);
  Height := MulDiv(Height, ARVStyle.UnitsPixelsPerInch, lpy);
end;
```

Example of using (making document width in editor equal to page width, taking difference in resolutions into account):

```
var w, h: TRVPixelLength(1085);

// rvoClientTextWidth must be excluded from RichViewEdit1.Options(247).

GetPageSize(RVPrint1, w,h);
RichViewEdit1.MaxTextWidth(246) :=
  w - RichViewEdit1.LeftMargin(245) - RichViewEdit1.RightMargin(251);
```

```
RichViewEdit1.MinTextWidth[246] := RichViewEdit1.MaxTextWidth[246];
RichViewEdit1.Format[295];
```

9.10 RVGetFirstEndnote and Others

These functions allow to enumerate endnotes[184], footnotes[185], sidenotes[186] and text box items[188] in a document.

Unit [VCL/FMX] RVNote / fmxRVNote.

```
function RVGetFirstEndnote(RichView: TCustomRichView[216];
  AllowHidden: Boolean): TRVEndnoteItemInfo[985];
function RVGetNextEndnote(RichView: TCustomRichView[216];
  Endnote: TRVEndnoteItemInfo[985];
  AllowHidden: Boolean): TRVEndnoteItemInfo[985];

function RVGetFirstFootnote(RichView: TCustomRichView[216];
  AllowHidden: Boolean): TRVFootnoteItemInfo[988];
function RVGetNextFootnote(RichView: TCustomRichView[216];
  Footnote: TRVFootnoteItemInfo[988];
  AllowHidden: Boolean): TRVFootnoteItemInfo[988];
```

(introduced in version 10)

Unit RVSidenote.

```
function RVGetFirstSidenote(RichView: TCustomRichView[216];
  AllowHidden: Boolean): TRVSidenoteItemInfo[990];
function RVGetNextSidenote(RichView: TCustomRichView[216];
  Sidenote: TRVSidenoteItemInfo[990];
  AllowHidden: Boolean): TRVSidenoteItemInfo[990];
```

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[106] notes/text boxes (see *rvepHidden* property[1067]). To enumerate only visible notes/text boxes, use *rvoShowHiddenText* in **RichView**.Options[247] as a value for this parameter.

9.11 RVGetNoteTextStyleNo

Returns index of text style for endnote[184] or footnote[184].

Unit [VCL/FMX] RVNote / fmxRVNote.

```
function RVGetNoteTextStyleNo(RVStyle: TRVStyle[739]; StyleNo: Integer;
  StyleTemplateName: TRVStyleTemplateName[1094]=''): Integer;
```

(introduced in version 10, changed in version 14)

This function returns text style (index in **RVStyle**.TextStyles[764]) which can be used for footnote or endnote characters.

If **StyleTemplateName** is not empty, and a style template with this name[845] exists in **RVStyle**.StyleTemplates[762], 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[764] [StyleNo], but SubSuperScriptType[818] = *rvsssSuperScript*.

In all cases, Protection[815] of the returned text style is [*rvprDoNotAutoSwitch*].

If such text style does not exist, this function adds it to the end of **RVStyle**.TextStyles[764].

The recommended names of style templates to use with this function:

- 'endnote reference' for endnotes[184] and note references[190] inside endnote text[984];
- 'footnote reference' for footnotes[185] and note references[190] inside footnote text[984].

9.12 RVIsURL, RVIsEmail

These function return *True* if the string is URL or e-mail address.

Unit [VCL/FMX] RVFileFuncs / fmxRVFileFuncs.

```
function RVIsURL(const s: TRVUnicodeString[1099]): Boolean;
function RVIsEmail(const s: TRVUnicodeString[1099]): Boolean;
```

(introduced in version 10; changed in version 18)

First, RVIsURL calls RVIsCustomURL[1126]. 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[138],
- to find which links are local links when loading RTF, DocX, Markdown, HTML files,
- to detect URLs in messages in TRVChat[640]
- in demo projects and in RichViewActions.

See also: how to create a chat window[1139].

9.13 RVMathEquationManager

Returns the object containing global properties for math equations [193].

Unit RVCustomMathWrapper.

```
function RVMathEquationManager: TRVCustomMathEquationManager[1020];
```

(introduced in version 23)

9.14 RVItemFactory

Returns the object used to creates items in documents [163].

Unit RVCustomMathWrapper.

```
function RVItemFactory: TRVRVItemFactory[1036];
```

(introduced in version 24)

You will rarely need to use this object directly. Typically, items in documents are created automatically by TRichView's methods and functions.

Even more rarely would you need to override the classes used for these items.

9.15 RVOpenURL

Opens URL in an external browser.

Unit [VCL/FMX] RVFileFuncs / fmxRVFileFuncs.

```
procedure RVOpenURL(const URL: String);
```

(introduced in version 20)

9.16 Unicode Conversion

Various Unicode [135]-related functions.

Unit [VCL / FMX] RVUni / fmxRVUni.

```
type
  TRVCodePage = type cardinal; // defined in RVStyle unit
function RVU_AnsiToUnicode(CodePage: TRVCodePage;
  const s: TRVAnsiString[1056]): TRVUnicodeString[1099];
function RVU_UnicodeToAnsi(CodePage: TRVCodePage;
  const s: TRVUnicodeString[1099]): TRVAnsiString[1056];
function RVU_AnsiToUTF8(CodePage: TRVCodePage;
  const s: TRVAnsiString[1056]): TRVRawByteString[1085];
```

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[1085]): 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[135].

10 Types

Use the tree navigation to view information about types.

10.1 TCheckpointData

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TCheckpointData = type Pointer;
```

Value of this type identifies one *checkpoint*[92].

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"[92] for details.

10.2 TRVAnsiString

Unit [VCL/FMX] RVTypes / fmxRVTypes;

Declaration for Lazarus[160], 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 (even in Lazarus[160])

See also

- TRVUnicodeString[1099]
- TRVRawByteString[1085]

10.3 TRVBackgroundPropertyId

Unit [VCL/FMX] RVBack / fmxRVBack;

```
type
  TRVBackgroundPropertyId = (
    rvbpidHorizontalAlignment,
    rvbpidVerticalAlignment,
    rvbpidHorizontalSizeKind,
    rvbpidVerticalSizeKind,
    rvbpidHorizontalOffsetKind,
    rvbpidVerticalOffsetKind,
    rvbpidHorizontalOffset,
    rvbpidVerticalOffset,
    rvbpidImageWidth,
    rvbpidImageHeight,
    rvbpidAttachment,
    rvbpidHorizontalRepeat,
    rvbpidVerticalRepeat,
    rvbpidVisible);
```

Identifies properties of TRVBackgroundProperties[1021].

Value	Property
rvbpidHorizontalAlignment	HorizontalAlignment[1023]
rvbpidVerticalAlignment	VerticalAlignment[1024]
rvbpidHorizontalSizeKind	HorizontalSizeKind[1024]
rvbpidVerticalSizeKind	VerticalSizeKind[1025]
rvbpidHorizontalOffsetKind	HorizontalOffsetKind[1023]

Value	Property
rvbpidVerticalOffsetKind	VerticalOffsetKind[1025]
rvbpidHorizontalOffset	HorizontalOffset[1023]
rvbpidVerticalOffset	VerticalOffset[1025]
rvbpidImageWidth	ImageWidth[1024]
rvbpidImageHeight	ImageHeight[1025]
rvbpidAttachment	Attachment[1022]
rvbpidHorizontalRepeat	HorizontalRepeat[1024]
rvbpidVerticalRepeat	VerticalRepeat[1025]
rvbpidVisible	Visible[1025]

This type is used in methods that assign properties of background as edition operations.

10.4 TRVBackgroundRepeat

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVBackgroundRepeat = (
    rvbrpRepeat,
    rvbrpNoRepeat,
    rvbrpSpace
  );
```

Specifies whether the background image is repeated, and how it is repeated.

Value	Meaning
rvbrpRepeat	The background image is repeated (tiled)
rvbrpNoRepeat	The background image is drawn only once, without repetition.
rvbrpSpace	The background image is repeated as many times as possible without being clipped, leaving equal spacing between the images

This type is used in:

- TRVBackgroundProperties[1021].HorizontalRepeat[1024]

10.5 TRVBackgroundSize

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVBackgroundSize = (
    rvbszAuto, rvbszContain,
    rvbszCover, rvbszValue);
```

The property specifies the background image size.

Value	Meaning
rvbszAuto	The size is adjusted proportionally to the opposite dimension. If the both dimensions are *rvbszAuto*, the original image size is used.
rvbszContain	Scales the background image so that the entire image fits within the background rectangle while preserving its aspect ratio
rvbszCover	Scales the background image so that the entire background rectangle is covered while preserving the aspect ratio
rvbszValue	Scales the background image to the specified size

This type is used in:

- TRVBackgroundProperties[1021].HorizontalSizeKind[1024]
- TRVBackgroundProperties[1021].VerticalSizeKind[1025]

10.6 TRVBiDiMode

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVBiDiMode =
    (rvbdUnspecified, rvbdLeftToRight, rvbdRightToLeft);
```

Value of this type defines default text reading direction for document (TRichView.BiDiMode[230]), paragraph (TParaInfo.BiDiMode[831]), or text (TFontInfo.BiDiMode[808]).

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.

Value	Meaning
rvbdRightToLeft	Default text reading order is right-to-left.

See also:

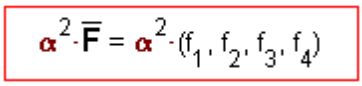
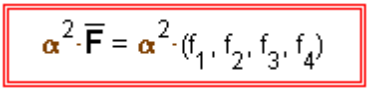
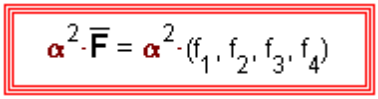
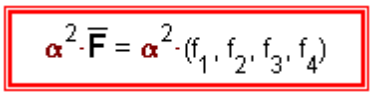
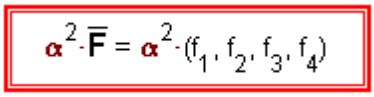
Bidirectional text in TRichView [137]

10.7 TRVBorderStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVBorderStyle = (rvbNone, rvbSingle, rvbDouble, rvbTriple,
    rvbThickInside, rvbThickOutside);
```

Style of border around paragraph [102] or around a text box (of a sidenote [186] or a text box item [188]).
Type for Border [831].Style [854] property.

Style	Border
rvbNone	No border
rvbSingle	
rvbDouble	
rvbTriple	
rvbThickInside	
rvbThickOutside	

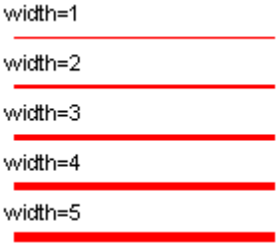
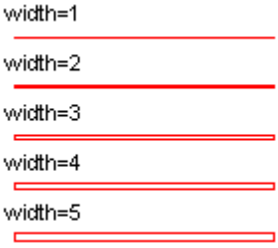
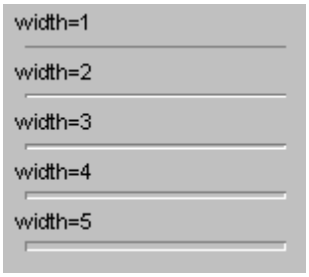
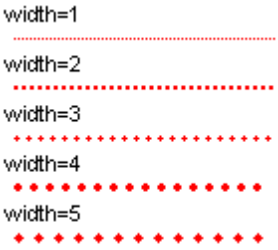
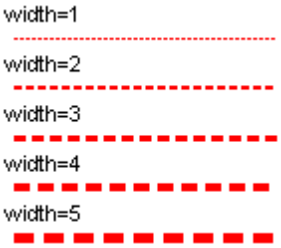
(examples are made with Color [853]=*clRed*, Width [854]=1, InternalWidth [853]=1)

10.8 TRVBreakStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVBreakStyle =
    (rvbsLine, rvbsRectangle,
     rvbs3d, rvbsDotted, rvbsDashed);
```

Style of *break* [173] item.

Style	Example
rvbsLine	
rvbsRectangle	
rvbs3d	 (break color is ignored)
rvbsDotted	
rvbsDashed	

10.9 TRVCellVAlign

Unit [VCL/FMX] RVTable / fmxRVTable;

```
type
  TRVCellVAlign =
    (rvcTop,  rvcMiddle, rvcBottom, rvcVDefault);
```

Types vertical alignment in cells.

See VAlign[968] property of table rows and VAlign[963] property of cells.

10.10 TRVChatFileType

A type of an attached file in TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat;

```
type
  TRVChatFileType = (
    rvcftDownload,
    rvcftDocument,
    rvcftImage,
    rvcftAudio,
    rvcftVideo,
    rvcftArchive
  );
```

Value	Meaning
rvcftDownload	The file is not downloaded yet
rvcftDocument	Document
rvcftImage	Image file
rvcftAudio	Sound file
rvcftVideo	Video file
rvcftArchive	Compressed archive

10.11 TRVChatItemId

Unit [VCL/FMX] RVChat / fmxRVChat;

```
type
  TRVChatItemId = TRVAnsiString[1056];
```

The type for identifiers of chat[640] messages and users.

10.12 TRVChatMessageState

Unit [VCL/FMX] RVChat / fmxRVChat;

Chat[640] message status.

```
type
  TRVChatMessageState = (
    rvcmsDefault, rvcmsNew, rvcmsSent,
    rvcmsRead, rvcmsFailed, rvcmsDeleted
  );
```

Value	Meaning
rvcmsDefault	no special status (for messages of other users)
rvcmsNew	new message
rvcmsSent	sent message
rvcmsRead	sent and read message
rvcmsFailed	failed to send
rvcmsDeleted	message marked as deleted

10.13 TRVChatQuoteType

A type of a quote of a message in TRVChat[640].

Unit [VCL/FMX] RVChat / fmxRVChat;

```
type
  TRVChatQuoteType = (rvcqtMessage, rvcqtQuotation);
```

Value	Meaning
rvcqtMessage	The entire message is quoted, though only a fragment of its text may be displayed (see Chat.Quote[643].MaxAutoQuoteLength[680]). If the quoted message is edited, the quote is automatically updated.
rvcqtQuotation	A fragment of the message is quoted. It will remain unchanged even if the original message is edited.

10.14 TRVCodePage

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVCodePage = type Cardinal;
```

A type for code page identifiers.

Some important values:

- 0 (CP_ACP) – ANSI code page of the default language in Windows
- 65001 (CP_UTF8) – Unicode in UTF-8 encoding
- $FFFFFFFF (RV_CP_DEFAULT[1118]) can be used only in several places to specify the default code page (which may be different depending on file format or on OS).

10.15 TRVColor

Unit [VCL/FMX] RVTypes / fmxRVTypes;

Color type.

Syntax (VCL and LCL)

```
type
  TRVColor = TColor;
```

Syntax (FireMonkey)

```
type
  TRVColor = TAlphaColor;
```

See also:

- Color constants[1105]

10.16 TRVColorMode

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVColorMode = (rvcmColor, rvcmVCLThemeColor*,
    rvcmFMXThemeColor**,
    rvcmPrinterColor,
    rvcmGrayScale, rvcmBlackAndWhite, rvcmBlackOnWhite);
```

* defined in VCL and LCL

** defined in FireMonkey

Mode	Meaning
rvcmColor	Color output.
rvcmVCLThemeColor [VCL and LCL]	If a custom function for applying system colors[1125] is defined, it is used. Otherwise, for Delphi XE2+, use system colors of the active VCL theme instead of Windows system colors. Otherwise, works as *rvcmColor*.
rvcmFMXThemeColor [FMX]	Use a foreground color defined in the component's FireMonkey style instead of

Mode	Meaning
	black color. Other colors are not affected.
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* [173], list markers [180], tables [179].

It does not affect drawing and printing pictures.

10.17 TRVCoord, TRVCoordPoint, TRVCoordSize, TRVCoordRect

Unit [VCL/FMX]: RVTypes / fmxRVTypes.

Types for coordinates: a single coordinate, a point, a size, a rectangle.

Syntax (VCL and LCL)

```
type
  TRVCoord = Integer;
  TRVCoordPoint = TPoint;
  TRVCoordSize = TSize;
  TRVCoordRect = TRect;
```

Syntax (FireMonkey)

```
type
  TRVCoord = Single;
  TRVCoordPoint = TPointF;
  TRVCoordSize = TSizeF;
  TRVCoordRect = TRectF;
```

10.18 TRVDBFieldFormat

Format for saving data in the database field

Unit [VCL/FMX] RichView / fmxRichView;

```
type
  TRVDBFieldFormat =
```

```
    (rvdbRVF, rvdbRTF, rvdbDocX, rvdbText, rvdbHTML,
     rvdbMarkdown)
```

Value	Meaning
rvdbRVF	RichView Format (RVF) [129]*
rvdbRTF	Rich Text Format (RTF)
rvdbDocX	Microsoft Word Document (DocX)*
rvdbText	Plain text**
rvdbHTML	HTML***
rvdbMarkdown	Markdown**

* RVF or DocX cannot be saved in WideMemo fields (fields having DataType=*ftWideMemo*). For these fields, RTF is saved.
** Unicode (UTF-16) text for WideMemo fields, text in the specified code page otherwise.
*** For WideMemo fields, HTML is saved in UTF-16 encoding, ignoring HTMLSaveProperties [244].Encoding [439] and EncodingStr [439].

See also:

- TRichView [216].Document [238].FieldFormat [431] (live binding)
- TDBRichViewEdit [616].FieldFormat [623]

10.19 TRVDisplayOptions

Unit [VCL/FMX] RVScroll / fmxRVScroll;

```
type
  TRVDisplayOption = (rvdoImages, rvdoComponents, rvdoBullets);
  TRVDisplayOptions = set of TRVDisplayOption;
const
  rvdoALL = [rvdoImages, rvdoComponents, rvdoBullets];
```

10.20 TRVDocRotation

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
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°

10.21 TRVDualLength, TRVDualSignedLength, TRVDualSignedLengthKind

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVDualSignedLengthKind = (rvdslkAbsolute, rvdslkPercent);
  TRVDualSignedLength = type Integer;
  TRVDualLength = type Integer;
```

TRVDualSignedLength value depends on the corresponding TRVDualSignedLengthKind:

- If *rvbpkAbsolute*: the same as TRVStyleLength[1094]
- If *rvbpkPercent*: 1/1000 of percent; for example, 50000 = 50%

TRVDualLength may contain only zero or positive values.

- If >= 0: the same as TRVStyleLength[1094]
- If < 0: -1/1000 of percent; for example, -50000 = 50%

10.22 TRVEnumScope

Unit [VCL/FMX]: RVWordEnum / fmxRVWordEnum;

A word enumeration scope.

Syntax

```
type
  TRVEnumScope = (rvesFromCursor, rvesFromStart);
```

Value	Meaning
rvesFromCursor	From the caret position to the end of the document
rvesFromStart	From the beginning to the end of the document

10.23 TRVESearchOptions

Unit [VCL/FMX] RVEdit / fmxRVEdit;

```
type
  TRVESearchOption = (rvseoMatchCase, rvseoDown, rvseoWholeWord,
    rvseoMultiItem, rvseoSmartStart, rvseoFromStored);
  TRVESearchOptions = set of TRVESearchOption;
```

Options for searching in TRichViewEdit, see TCustomRichViewEdit.SearchText[553].

Value	Meaning
rvseoMatchCase	If included, character case is taken into account when comparing string.

Value	Meaning
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 [167]. 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) See also *rvseoFromStored*.
rvseoFromStored	If included, searching starts from the position of the last stored search result [375] instead of the current selection or caret position. This option affects only cases when searching starts from a selection.

10.24 TRVExtraItemProperty

Unit [VCL/FMX] RVItem / fmxRVItem;

```
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.

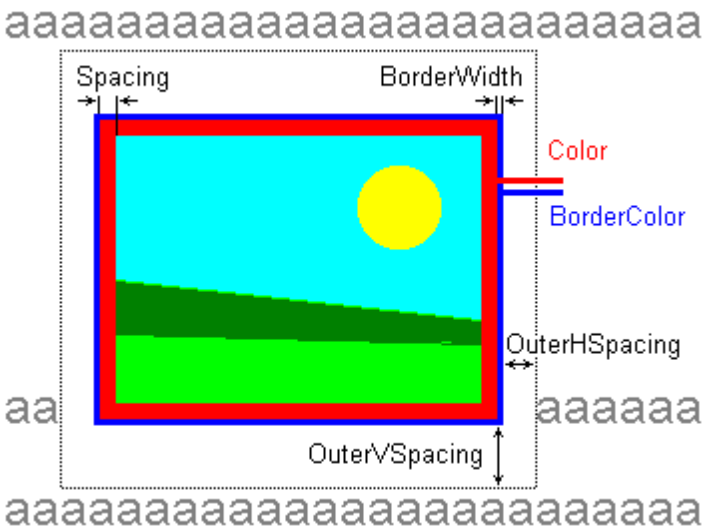
Positioning

Property	Applicable To	Default Value	Meaning
rvepVShift	rect.items*	0	Vertical offset, in % of item height (default), or in RVStyle.Units [764] (see *rvepVShiftAbs*). Negative values move the item down, positive move it up (relative to the position specified in VAlign [1100])
rvepVShiftAbs	rect.items*	0	If non-zero, *rvepVShift* defines vertical offset in RVStyle.Units [764] instead of %.
rvepVAlign	rect.items*	ord(*rvvaBaseline*)	Vertical alignment, TRVVAlign [1100] converted to integer.

* "rect.items" include pictures [169], *hot-pictures* [172], controls [174], *bullets* [176], *hotspots* [178], labels [181], sequences [183], footnotes [185], endnotes [184], note references [190].

Borders and spacing

Scheme:



Property	Applicable To	Default Value	Meaning
rvepSpacing	rect.items*	1	Spacing around the item content, in RVStyle.Units [764]. Must be a zero or a positive value.

Property	Applicable To	Default Value	Meaning
			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 (TRVColor[1063] value). It is drawn below the image (and around the image, if *rvepSpacing* is positive).
rvepBorderWidth	rect.items*	0	Border width, in RVStyle.Units[764]. Must be a zero or a positive value.
rvepBorderColor	rect.items*	*clBlack*	Border color (TRVColor[1063] value).
rvepOuterHSpacing *rvepOuterVSpacing*	rect.items*	0	Spacing around the border (horizontal and vertical), in RVStyle.Units[764]. Must be a zero or a positive value.

* "rect.items" include pictures[169], *hot-pictures*[172], controls[174], *bullets*[176], *hotspots*[178], labels[181], sequences[183], footnotes[185], endnotes[184], note references[190].

Pictures

Property	Applicable To	Default Value	Meaning
rvepImageWidth, *rvepImageHeight*	▪ pictures[169] ▪ *hot-pictures*[172]	0	If set to positive value, define size of the image in RVStyle.Units[764]. The actual image is not modified, it is just drawn stretched. See also *rvepAnimationInterval* property below.
[VCL and LCL] *rvepTransparent*, *rvepTransparentMode*, *rvepTransparentColor*	▪ pictures[169] ▪ *hot-pictures*[172] (if they are TBitmap)		Correspond to properties of TBitmap: Transparent, TransparentMode, TransparentColor. Limitations: ▪ 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)

Property	Applicable To	Default Value	Meaning
rvepMinHeightOnPage	▪ pictures [169] ▪ *hot-pictures* [172]	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 [887] 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 [169] ▪ *hot-pictures* [172] ▪ *bullets* [176] ▪ *hotspots* [178] ▪ list markers [180]	0	If non-zero, image size is not saved in HTML, even if *rvhtmlsioImageSizes* is included in HTMLSaveProperties [244].ImageOptions [442]. This option is ignored for pictures [169] and *hot-pictures* [172], if *rvepImageWidth* or *rvepImageHeight* properties are non-zero.
rvepAnimationInterval	▪ pictures [169] ▪ *hot-pictures* [172]	0	Animation delay in 1/100 of second. If nonzero and *rvepImageWidth* and/or *rvepImageHeight* are defined, playing grid animation [124] (in imagewidth x imageheight frame) This feature works only if RVGridAnimate unit is included in the project.
rvepSmoothScaling	▪ pictures [169] ▪ *hot-pictures* [172]	1	If non-zero, TRichView uses smooth resizing method for pictures.

Property	Applicable To	Default Value	Meaning
			If zero, TRichView uses fast resizing method, and resized images look pixelated. Good for images like barcodes and pixel art.

Editing

Property	Applicable To	Default Value	Meaning
rvepResizable	▪ pictures [169] ▪ *hot-pictures* [172] ▪ controls [174]	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 [815] for text style.

Other

Property	Applicable To	Default Value	Meaning
rvepVisible	▪ controls [174]	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 [106].
rvepShared	▪ pictures [169] ▪ *hot-pictures* [172] ▪ controls [174]	0	If non-zero, the item content (picture or control) is not deleted when the item is deleted.

See also

- TRVExtraItemStrProperty [1072]

See also methods of TRichView

- SetItemExtraIntProperty [366],
- GetItemExtraIntProperty [307].

See also methods of TRichViewEdit

- SetItemExtraIntPropertyEd [572],
- SetCurrentItemExtraIntProperty [566],

- GetCurrentItemExtraIntProperty[518].

10.25 TRVExtraItemStrProperty

Unit [VCL/FMX] RVItem / fmxRVItem;

```
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[169] ▪ *hot-pictures*[172] ▪ *bullets*[176] ▪ *hotspots*[178]	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[247] and RichView.ShowHint=*True*. Hints are displayed by assigning to RichView.Hint. See also OnItemHint[388] 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[388]. 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[418] event, the proper value of Extras parameter is set by default.
rvespImageFileName	▪ pictures[169] ▪ *hot-pictures*[172] ▪ tables[179]	Provides a place for storing image file name. For tables, it is a background image, also accessible as table.BackgroundImageFileName[905].

Property	Applicable To	Meaning
		See also: TRVTableCellData.BackgroundImageFileName [953] f *rvhtmlsioUseItemImageFileNames* is included in HTMLSaveProperties [244].ImageOptions [442], 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 [247].
rvespTag	all items	tag [96]

See also

- TRVExtraItemProperty [1067]

See also methods of TRichView

- SetItemExtraStrProperty [367],
- GetItemExtraStrProperty [308].

See also methods of TRichViewEdit

- SetItemExtraStrPropertyEd [573],
- SetCurrentItemExtraStrProperty [567],
- GetCurrentItemExtraStrProperty [519].

10.26 TRVFloatPosition

Unit [VCL/FMX] RVFloatingPos / fmxRVFloatingPos;

```
type
  TRVFloatPosition = type Integer;
```

This is a type of HorizontalOffset [999] and VerticalOffset [1003] properties of TRVBoxPosition [997].

Meaning of these values depend on HorizontalPositionKind [999] and VerticalPositionKind [1003] properties, and may mean either values in TRVStyleUnits [1094] or in (1/1000)%.

10.27 TRVFloatPositionKind

Unit [VCL/FMX] RVFloatingPos / fmxRVFloatingPos;

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 [997]:

- HorizontalPositionKind [999]

- VerticalPositionKind[1003]

Value	Meaning
rvfpkAlignment	A position is specified in HorizontalAlignment[999] /VerticalAlignment[1003]
rvfpkAbsPosition	HorizontalOffset[999]/VerticalOffset[1003] contains offset in RVStyle.Units[764]
rvfpkPercentPosition	HorizontalOffset[999]/VerticalOffset[1003] contains offset in (1/1000)% of the anchor area size

10.28 TRVFloatProperty

Unit [VCL/FMX] RichView / fmxRichView;

Identifies a property of TRichView[216] of a TRVLength[1081] type.

```
type
  TRVFloatProperty = (rvfpDPLeftMargin, rvfpDPRightMargin,
    rvfpDPTopMargin, rvfpDPBottomMargin,
    rvfpDPHeaderY, rvfpDPFooterY);
```

Value	Corresponding property of TRichView
rvfpDPLeftMargin	DocParameters[237].LeftMargin[425]
rvfpDPRightMargin	DocParameters.RightMargin[426]
rvfpDPTopMargin	DocParameters.TopMargin[427]
rvfpDPBottomMargin	DocParameters.BottomMargin[424]
rvfpDPHeaderY	DocParameters.HeaderY[425]
rvfpDPFooterY	DocParameters.FooterY[425]

A parameter of this type is used in TRichViewEdit[469].SetFloatPropertyEd[555].

See also:

- TRVIntProperty[1080];
- TRVStrProperty[1093].

10.29 TRVFloatSize

Unit [VCL/FMX] RVFloatingBox / fmxRVFloatingBox;

```
type
  TRVFloatSize = type Integer;
```

This is a type of Width[996] and Height[995] properties of TRVBoxProperties[994].

Meaning of these values depend on WidthType[996] and HeightType[995] properties, and may mean either values in TRVStyleUnits[1094] or in (1/1000)%.

10.30 TRVFontCharset

Unit [VCL/FMX] RVTypes / fmxRVTypes;

TRVFontCharset represents the character set of a font.

Syntax (VCL and LCL)

```
type
  TRVColor = TFontCharset;
```

Syntax (FireMonkey)

```
type
  TRVFontCharset = type Byte;
```

10.31 TRVFontInfoProperties

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
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 [821].

This is a type of the following properties:

- TRVStyleTemplate.ValidTextProperties [848];
- TFontInfo.ModifiedProperties [824].

Parameters of this type are used in the following methods:

- TFontInfos.FindSuchStyle, FindSuchStyleEx [791];
- TFontInfo.IsEqual [827].

```
const
  RVAllFontInfoProperties: TRVFontInfoProperties =
    [Low(TRVFontInfoProperty)..High(TRVFontInfoProperty)];
```

Constant, lists all TFontInfo properties.

10.32 TRVFontSize

Unit [VCL/FMX]: RVTypes / fmxRVTypes;

Font size type.

Syntax (VCL and LCL)

```
type
  TRVFontSize = Integer;
```

Syntax (FireMonkey)

```
type
  TRVFontSize = Single;
```

See also:

- TCustomRVFontInfo[806].Size[817]

10.33 TRVFReaderStyleMode

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVFReaderStyleMode =
  (rvf_sIgnore, rvf_sInsertMap, rvf_sInsertMerge)
```

Defines a mode for loading collection of styles from RVF[129] (if they are contained there, see *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions[254]).

This is a type of the following properties of TCustomRichView:

- RVFTextStylesReadMode[258] (loading Style[260].TextStyles[764]);
- RVFParaStylesReadMode[258] (loading Style.ParaStyles[757] and Style.ListStyles[755]).

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[739] component is used exclusively by the given TRichView[Edit].

If style templates are used[263], and RVFTextStylesReadMode[258]=RVFParaStylesReadMode[258] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[260]<>*rvstimIgnoreSourceStyleTemplates*, the RVF loading methods replace the existing Style[260].StyleTemplates[762] 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[260].

Loading operations include:

- LoadRVF[336],
- LoadRVFFromStream[335],
- loading from database fields[604].

Insertion operations include:

- InsertRVFFromStream[323],
- AppendRVFFromStream[283],
- TRichViewEdit.InsertRVFFromFileEd[537],
- TRichViewEdit.InsertRVFFromStreamEd[538],
- pasting[113] from the Clipboard,
- drag&drop[123].

10.34 TRVGraphicType

Identifies a graphic format.

Unit [VCL/FMX]: RVTypes / fmxRVTypes

```
type
  TRVGraphicType =
    (rvgtBitmap, rvgtIcon, rvgtMetafile, rvgtJPEG, rvgtPNG,
     rvgtGIF, rvgtTIFF, rvgtAnyMap, rvgtSVG, rvgtWDP,
     rvgtWebP, rvgtOther);
```

Value	Meaning	Supported in frameworks*
rvgtBitmap	Windows bitmap	All
rvgtIcon	Windows icon	All
rvgtMetafile	Windows metafile, or Windows enhanced metafile	VCL
rvgtJPEG	JPEG image (Joint Photographic Experts Group)	All
rvgtPNG	PNG image (Portable Network Graphics)	All
rvgtGIF	GIF image (Graphics Interchange Format)	VCL, FMX
rvgtTIFF	TIFF image (Tagged Image File Format)	VCL, FMX
rvgtAnyMap	AnyMap image	LCL
rvgtSVG	SVG image (Scalable Vector Graphic)	VCL (requires third-party classes)

Value	Meaning	Supported in frameworks*
		FMX (requires Skia4Delphi)
rvgtWDP	JPEG XR (also known as HD Photo and Windows Media Photo). File extensions: HDP, JXR, WDP.	FMX
rvgtWebP	WebP image	FMX (requires Skia4Delphi)
rvgtOther	Other or unknown image format	

* even if a graphic format is not supported in by default, it may be implemented in third-party graphic classes (in VCL/LCL), or in third-party bitmap codecs (FMX).

In FireMonkey, availability of codecs for reading/writing image formats depends on the platform.

10.35 TRVHFType

Type of header or footer

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
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

10.36 TRVHTMLEncoding

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVHTMLEncoding =
    (rvhtmleUTF8, rvhtmleANSI);
```

Encoding for HTML saving

Value	Meaning
rvhtmleUTF8	UTF-8

Value	Meaning
rvhtmleANSI	ANSI (Windows) encoding, corresponding to TRVStyle[739].TextStyles[764][0].Charset[809]

10.37 TRVHTMLLength

Unit [VCL/FMX]: RVTable / fmxRVTable;

```
type
  TRVHTMLLength = type Integer;
```

Represents length either in percent or in TRVStyleUnits[1094].

Range	Meaning
>0	Length in TRVStyleUnits[1094]
<0	Length in percent (for example, -50 means 50%)
=0	Length is undefined (default)

See also

- TRVTableItemInfo.BestWidth[906]
- TRVTableCellData.BestWidth[955]

10.38 TRVHTMLSavingPart

Unit [VCL/FMX] RVStyle / fmxRVStyle;

Specifies which part of HTML must be saved.

```
type
  TRVHTMLSavingPart = (
    rvhtmlspComplete,
    rvhtmlspFirst,
    rvhtmlspMiddle,
    rvhtmlspLast);
```

Value	Meaning
rvhtmlspComplete	Complete HTML
rvhtmlspFirst	Opening part of HTML (including opening tag of <html>, <head>, and opening tag of <body>)
rvhtmlspMiddle	Content inside <body>
rvhtmlspLast	Closing part of HTML (including </body> and </html>)

10.39 TRVIntProperty

Unit [VCL/FMX] RichView / fmxRichView.

Identifies a property of TRichView[216] 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[245]
rvipRightMargin	RightMargin[251]
rvipTopMargin	TopMargin[262]
rvipBottomMargin	BottomMargin[231]
rvipMaxTextWidth	MaxTextWidth[246]
rvipMinTextWidth	MinTextWidth[246]
rvipBackgroundStyle	BackgroundStyle[229] (deprecated, use SetBackgroundPropertyEd[557])
rvipColor	Color[233]
rvipBiDiMode	BiDiMode[230]
rvipDPUnits	DocParameters[237].Units[428]
rvipDPOrientation	DocParameters.Orientation[426]
rvipDPTitlePage	DocParameters.TitlePage[427]
rvipDPFacingPages	DocParameters.FacingPages[424]
rvipDPMirrorMargins	DocParameters.MirrorMargin[426]

A parameter of this type is used in TRichViewEdit[469].SetIntPropertyEd[555].

See also:

- TRVFloatProperty[1074];
- TRVStrProperty[1093].

10.40 TRVItemBackgroundStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVItemBackgroundStyle =
```

```
    (rvbsColor, rvbsStretched, rvbsTiled,
     rvbsCentered, rvbsOther);
```

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
rvbsOther	The value specified in BackgroundProperties does not have an analog in TRVItemBackgroundStyle

See also:

- TRVTableItemInfo.BackgroundStyle[905] (background of table)
- TRVTableCellData.BackgroundStyle[954] (background of cell)
- Smooth image scaling[125]

10.41 TRVLength

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVLength = type Extended;
```

This is a type of values measured in TRVUnits[1099].

10.42 TRVLoadFormat

Unit [VCL/FMX] RVStyle / fmxRVStyle;

```
type
  TRVLoadFormat =
    (rvlfText, rvlfHTML, rvlfRTF, rvlfRVF,
     rvlfDocX, rvlfMarkdown, rvlfURL, rvlfOther);
```

Identifies a loading format.

See also:

- TRichView.OnReadHyperlink[395]

10.43 TRVOleDropEffects

Unit [VCL/FMX]: RVEdit / fmxRVEdit.

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[589];
- OnOleDragOver[590];
- OnOleDrop[591].

10.44 TRVOpacity

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
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.

10.45 TRVPageNumberType

Unit [VCL/FMX]: RVFieldItems / fmxRVFieldItems.

```
type
  TRVPageNumberType = (rvpntDefault, rvpntDecimal,
    rvpntDecimalDash,
    rvpntLowerAlpha, rvpntUpperAlpha,
    rvpntLowerRoman, rvpntUpperRoman);
```

Numbering type for "page number" field[190].

Type	Meaning
rvpntDefault	Default numbering. Displayed as decimal.
rvpntDecimal	1, 2, 3,...
rvpntDecimalDash	- 1 -, - 2 -, - 3 -

Type	Meaning
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[1009].

10.46 TRVPageSet

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVPageSet = (rvpsAll, rvpsOdd, rvpsEven);
```

Set of pages.

Value	Meaning
rvpsAll	All pages
rvpsOdd	Odd pages
rvpsEven	Even pages

See also:
- TRVPrint[696].Print[713]
- TRVPrint.PrintPages[714]

10.47 TRVParaInfoProperties

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
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[828].

This is a type of the following properties:

- TRVStyleTemplate.ValidParaProperties[847];
- TParaInfo.ModifiedProperties[839].

Parameters of this type are used in the following methods:

- TParaInfos.FindSuchStyle, FindSuchStyleEx[794];
- TParaInfo.IsEqual[841].

```
const
  RVAllParaInfoProperties: TRVParaInfoProperties =
    [Low(TRVParaInfoProperty)..High(TRVParaInfoProperty)];
```

A constant, lists all TParaInfo properties.

10.48 TRVPDFMetadata

A record containing information using when creating a PDF file.

Unit [VCL/FMX] PtblRV / fmxPtblRV;

```
type
  TRVPDFMetadata = record
    Title, Author, Subject, Keywords,
    Creator, Producer: TRVUnicodeString[1099];
    Creation, Modified: TDateTime;
    RasterDPI: Single;
    PDFA: Boolean;
    EncodingQuality: Integer;
  end;
  PRVPDFMetadata = ^TRVPDFMetadata;
```

10.49 TRVPenStyle

Unit [VCL/FMX]: RVTypes / fmxRVTypes.

A pen style type.

Syntax (VCL and LCL)

```
type
  TRVPenStyle = TPenStyle;
```

Syntax (FireMonkey)

```
type
  TRVPenStyle = (rvpsSolid, rvpsDash,
    rvpsDot, rvpsDashDot,
    rvpsDashDotDot, rvpsClear,
```

```
    rvpsInsideFrame);
```

See also:

- Pen style constants [1110]

10.50 TRVPixel96Length

Unit [VCL/FMX]: RVType / fmxRVType.

```
type
  TRVPixel96Length = type Integer;
```

This is a type of values measured in pixels, where 1 pixel = 1/96 of an inch.

Similar units are used as pixels in HTML.

See also:

Units of measurement in TRichView [144]

10.51 TRVPixelLength

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVPixelLength = type Integer;
```

This is a type of values measured in pixels, where 1 pixel = 1/TRVStyle [739].UnitsPixelsPerInch [765] of an inch.

See also:

Units of measurement in TRichView [144]

10.52 TRVPrintingStep

Unit [VCL/FMX]: PtblRV / fmxPtblRV.

```
type
  TRVPrintingStep = (rvpsStarting, rvpsProceeding, rvpsFinished);
```

Parameter of this type is used in OnSendingToPrinter [719] and OnFormatting [717] events.

10.53 TRVRawByteString

Unit [VCL/FMX]: RVTypes / fmxRVTypes.

Declaration for Lazarus [160], 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 [1056]

- TRVUnicodeString [1099]

10.54 TRVReaderStyleMode

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
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 [466] and ParaStyleMode [464].

10.55 TRVSaveFormat

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVSaveFormat =
  (rvsfText, rvsfHTML, rvsfRTF, rvsfRVF, rvsfDocX,
    rvsfMarkdown, rvsfXML);
```

Identifies a saving format.

Note: *rvsfXML* is used by TRichViewXML component.

This type is used in some events of TRichView, such as OnSaveComponentToFile [405] and OnSaveItemToFile [411], and SavePicture [349] method.

10.56 TRVSaveOptions

Deprecated. Use TRichView.HTMLSaveProperties [244].

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVSaveOption = (rvsoOverrideImages,
```

```
    rvsoFirstOnly, rvsoMiddleOnly, rvsoLastOnly,
    rvsoDefault0Style, rvsoNoHypertextImageBorders,
    rvsoImageSizes, rvsoForceNonTextCSS, rvsoUseCheckpointsNames,
    rvsoMarkersAsText, rvsoNoDefCSSStyle, rvsoInlineCSS,
    rvsoUseItemImageFileNames,
    rvsoXHTML, rvsoUTF8, rvsoHeaderFooter, rvsoInlineImages);
  TRVSaveOptions = set of TRVSaveOption;
```

Options for HTML export. Used for the parameter of deprecated HTML saving methods: SaveHTML[343], SaveHTMLEx[345], SaveHTMLToStream[347], SaveHTMLToStreamEx[348].

Options for saving images, both for SaveHTML[343], SaveHTMLEx[345]

Option	Meaning
rvsoOverrideImages	If set, TRichView can override existing images. If not set, TRichView will generate unique names for images.
rvsoNoHypertextImageBorders	Obsolete, does nothing
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[1067]. Sizes are always saved for such images. You can override this setting for the specific item using *rveoNoHTMLImagesSize* item extra integer properties[1067].
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[383] and OnSaveImage2[410] events have initial value of the Location parameter equal to the image file name. File names for items are defined via *rvespImageFileName* string property[1072].
rvsoInlineImages	If included, images are saved directly in HTML (in base64 encoding) instead of external files.

Options for saving list markers, both for SaveHTML[343], SaveHTMLEx[345]

Option	Meaning
rvsoMarkersAsText	If included, paragraph lists[180] 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[345]); CSS of lists are inserted directly in HTML:

- *rvlstBullet, rvlstUnicodeBullet*[864] – exported as bullets with default markers (disc, circle or square). FormatString[861] and Font[861] are ignored;
- *rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman*[864] – exported as lists with the proper numbering. FormatString[861] and Font[861] are ignored;
- *rvlstPicture,rvlstImageList,rvlstImageListCounter*[864] – exported as a bullet with picture. It's possible to change default image saving by OnHTMLSaveImage[383];
- indents are saved, but not very accurately.

Export without CSS (SaveHTML[343]):

- *rvlstBullet, rvlstUnicodeBullet,rvlstPicture,rvlstImageList*[864] – exported as bullets with default markers (disc, circle or square). FormatString[861], Picture[868], ImageList[863] and Font[861] are ignored;
- *rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman*[864] – exported as lists with the proper numbering. FormatString[861] and Font[861] are ignored;
- *rvlstImageListCounter*[864] – 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[861] and format strings[861] settings are respected.

Saving without CSS (SaveHTML[343]): all indents are ignored.

Saving with CSS (SaveHTMLEx[345]): LeftIndent[864] and MarkerIndent[866] are retained. FirstIndent[860] 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[343], SaveHTMLEx[345]

Option	Meaning
rvsoUseCheckpointsNames	*Checkpoints*[92]' names are used instead of checkpoint indices. Recommended.
rvsoXHTML	If included, XHTML is saved instead of HTML

Option	Meaning
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[764][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.
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[363] /SetFooter[362] methods. If DocParameters[237].TitlePage[427] = *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 SaveHTML[343]

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[345]

Option	Meaning
rvsoNoDefCSSStyle	By default, SaveHTMLEx assigns attributes of TextStyles[764][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[757][0].

Option	Meaning
	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[343], SaveHTMLEx[345]

These options are useful when generating HTML documents what will be inserted in larger HTML document.

Option	Meaning
rvsoFirstOnly	If set, TRichView saves only opening part of HTML (opens <html> tag, saves full <head> tag, opens <body> tag)
rvsoMiddleOnly	If set, TRichView saves only body of HTML document (content inside <body> tag)
rvsoLastOnly	If set, TRichView 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).

10.57 TRVScreenAndDevice, PRVScreenAndDevice

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVScreenAndDevice = record
    ppixScreen, ppiyScreen,
    ppixDevice, ppiyDevice,
    ppiDocument: Integer;
  end;
  PRVScreenAndDevice = ^TRVScreenAndDevice;
```

This record stores information necessary to convert values to/from device (such as printer) pixels.

ppixScreen, ppiyScreen – basic DPI, usually TRVStyle[739].UnitsPixelsPerInch[765] (they are called "screen" for historical reasons)

ppixDevice – number of pixels per inch along the device width.

ppiyDevice – number of pixels per inch along the device height.

ppiDocument – number of pixels per inch in TRichViiew (see DocumentPixelsPerInch[239]).

This structure is filled automatically, you do not need to use its members.

10.58 TRVSearchOptions

Unit [VCL/FMX]: RVScroll / fmxRVScroll.

```
type
  TRVSearchOption = (rvsroMatchCase, rvsroDown, rvsroWholeWord,
    rvsroFromStart, rvsroMultiItem, rvsroSmartStart,
    rvsroFromStored);
  TRVSearchOptions = set of TRVSearchOption;
```

Options for searching in RichView, see TCustomRichView.SearchText[354].

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. See also *rvsroFromStored*.
rvsroMultiItem	If included, the search can match substrings of several text items[167]. 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.

Value	Meaning
	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). See also *rvsroFromStored*.
rvsroFromStored	If included, searching starts from the position of the last stored search result[375] instead of the current selection. This option affects only cases when searching starts from a selection (but if you use this feature, include this option even in the first search that has *rvsroFromStart* option).

10.59 TRVSeqType

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVSeqType = (rvseqDecimal, rvseqLowerAlpha,
    rvseqUpperAlpha, rvseqLowerRoman, rvseqUpperRoman);
```

Numbering type for sequences[183], endnotes[184], footnotes[185], sidenotes[186].

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[746];
- TRVStyle.FootnoteNumbering[746];
- TRVStyle.SidenoteNumbering[746];
- TRVSeqItemInfo.NumberType[980].

10.60 TRVSpellFormAction

Unit [VCL/FMX] RVSpellChecker / fmxRVSpellCheckerWin;

```
type
  TRVSpellFormAction = (rvsfaIgnore, rvsfaIgnoreAll,
    rvsfaChange, rvsfaChangeAll, rvsfaAdd,
    rvsfaGuessCompletions);
  TRVSpellFormActions = set of TRVSpellFormAction;
```

Defines operations that can be performed by a spelling checker.

Value	Meaning
rvsfaIgnore	Ignore a single occurrence of a word
rvsfaIgnoreAll	Ignore all occurrences of a word
rvsfaChange	Change one occurrence of a word to another word
rvsfaChangeAll	Change all occurrences of a word to another
rvsfaAdd	Add a word to the dictionary.
rvsfaGuessCompletions	Returns possible completions of a word

See also:

- TRVSpellChecker[728].OnSpellFormAction[738]
- TRVSpellChecker[728].OnSpellFormAction[738]

10.61 TRVStrProperty

Unit [VCL/FMX]: RichView / fmxRichView.

Identifies a property of TRichView[216] of a string type.

```
type
  TRVStrProperty = (rvspDPTitle, rvspDPAuthor, rvspDPComments);
```

Value	Corresponding property of TRichView
rvspDPTitle	DocParameters[237].Title[427]
rvspDPAuthor	DocParameters.Author[424]
rvspDPComments	DocParameters.Comments[424]

A parameter of this type is used in TRichViewEdit[469].SetStrPropertyEd[555].

See also:

- TRVFloatProperty[1074];
- TRVIntProperty[1080].

10.62 TRVStyleLength

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

```
type
  TRVStyleLength = type Integer;
```

This is a type of values measured in TRVStyleUnits[1094] (i.e. pixels, twips or EMU).

See also:

Units of measurement in TRichView[144]

10.63 TRVStyleTemplateId

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVStyleTemplateId = type Integer;
```

Values of this type identify items in TRVStyleTemplateCollection[797],

This is a type of the following properties:

- Id[844] of TRVStyleTemplate[843] (unique style template identifier);
- ParentId[847] of TRVStyleTemplate[843] (a reference to the parent style template);
- NextId[846] of TRVStyleTemplate[843] (a reference to the style template for following paragraphs);
- StyleTemplateId[805] of text and paragraph attributes (styles), linking them to a style template.
- ParaStyleTemplateId[815] 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.

10.64 TRVStyleTemplateName

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVStyleTemplateName = type TRVUnicodeString[1099];
```

(changed in version 18)

This is a type of the following properties:

- Name of TRVStyleTemplate[843] (unique style template name)

10.65 TRVStyleUnits

Unit [VCL/FMX]: RVStyle/fmxRVStyle.

```
type
  TRVStyleUnits = (rvstuPixels, rvstuTwips, rvstuEMU);
```

This type is used to identify measure units that used internally in a program.

Value	Meaning
rvstuPixels	Pixels.

Value	Meaning
	1 pixel = 1/TRVStyle[739].UnitsPixelsPerInch[765] of an inch (normally, 1/96 of an inch)
rvstuTwips	Twips. 1 twip = 1/20 of a point = 1/1440 of an inch
rvstuEMU	English metric units. 1 EMU = 1/914400 of an inch = 1/36000 mm

Values measured in TRVUnits have type TRVStyleLength[1094].

Pixels

This is a type of TRVStyle[739].Units[764]. If it is equal to *rvstuPixels*, values are measured in pixels with DPI specified in TRVStyle.UnitsPixelsPerInch[765]; when rendering on a device (screen or printer), lengths are recalculated according to the device DPI:

<length on device> = <value measured in TRVStyleUnits> * <device DPI> div UnitsPixelsPerInch.

If you do not change value of UnitsPixelsPerInch, pixels are absolute measure units: lengths measured in them do not depend on the device DPI (like twips and EMUs).

However, if you always assign Screen.PixelsPerInch to UnitsPixelsPerInch, pixels become relative units (actual lengths of sizes measured in pixels depend on the screen DPI).

Examples

The table below shows how values measured in standard units are converted to various TRVStyleUnits.

Sample Value	Pixels (DPI = 96)	Twips	EMUs
1 inch	96	1440	914400
1 mm	≈ 3.78	≈ 56.69	36000
1 point	≈ 1.33	20	12700
1 pixel (DPI = 96)	1	15	9525

It's recommended to use *rvstuPixels* if sizes are displayed to users as pixels as well. Otherwise, it's recommended to use twips or EMUs.

Twips are OK if sizes are displayed as inches or points, and you do not need precision higher than 0.1.

EMUs are OK in all cases. They provide precision 0.01 for inches and points, 0.001 for mm, 1 for pixels.

See also:

Units of measurement in TRichView [144]

10.66 TRVTableBorderStyle

Unit [VCL/FMX] RVTable / fmxRVTable;

```
type
  TRVTableBorderStyle =
    (rvtbRaised, rvtbLowered, rvtbColor,
     rvtbRaisedColor, rvtbLoweredColor);
```

Defines style of border in table. This is a type of BorderStyle [908] and CellBorderStyle [910] properties of table.

Value	Meaning
rvtbRaised	Raised 3d border, with default colors.
rvtbLowered	Sunken 3d border, with default colors.
rvtbColor	Flat border of BorderColor [906]/CellBorderColor [909] color.
rvtbRaisedColor	Raised 3d border: ▪ left and top sides: BorderLightColor [907] /CellBorderLightColor [910]; ▪ bottom and right sides: BorderColor [906] /CellBorderColor [909]).
rvtbLoweredColor	Sunken 3d border: ▪ left and top sides: BorderColor [906]/CellBorderColor [909]); ▪ bottom and right sides: BorderLightColor [907] /CellBorderLightColor [910]).

See also

- Colors and layout of tables [199]

10.67 TRVTag

Unit [VCL/FMX] RVStyle / fmxRVStyle.

```
type
  TRVTag = type TRVUnicodeString[1099];
```

This is the type of item tags – additional value associated with any item [163] in TRichView [216] document, and with any table cell [963].

See also:

- Tags [96]

10.68 TRVTextDrawStates

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

```
type
  TRVTextDrawState = (rvtsSelected, rvtsHover,
    rvtsItemStart, rvtsItemEnd,
    rvtsDrawItemStart, rvtsDrawItemEnd,
    rvtsControlFocused, rvtsSpecialCharacters);
  TRVTextDrawStates = set of TRVTextDrawState;
```

States of text, used for drawing.

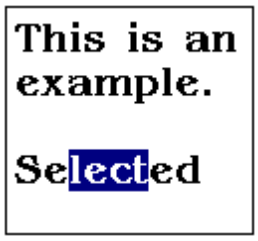
See:

- TRVStyle.OnDrawStyleText [781], OnDrawTextBack [783];
- TFontInfo.Draw [825]

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 [247].

* 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[781] 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*]).

10.69 TRVUndoType

Unit [VCL/FMX]: RVEdit / fmxRVEdit.

```
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*[96]) can have special meaning for an application (editing a hyperlink or another item property).

The demo Demos*\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[504]).

10.70 TRVUnicodeString

Unit [VCL/FMX]: RVTypes / fmxRVTypes

Declaration for Lazarus[160], 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[1056]
- TRVRawByteString[1085]

10.71 TRVUnits

Unit [VCL/FMX] RVStyle / fmxRVStyle.

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 of an inch = 12 points.
rvuPixels	Pixels. 1 pixel = 1/TRVStyle[739].UnitsPerInch[765] of an inch. (normally, 1/96 of an inch)
rvuPoints	Points (used in typography). 1 point = 1/72 of an inch.

This is a type of the following properties:

- TRichView[216].DocParameters[237].Units[428].
- TRVPrint[696].Units[707]

Values measured in TRVUnits have the type TRVLength[1081].

See also:

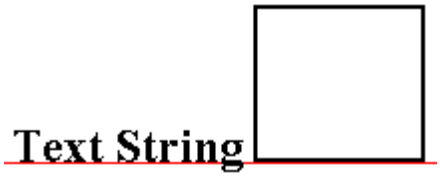
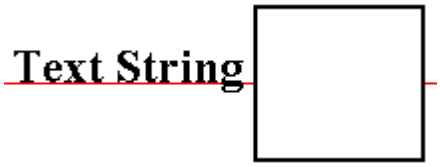
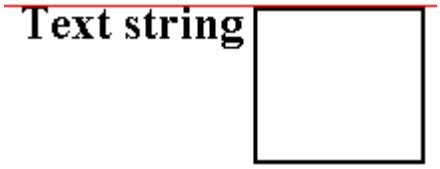
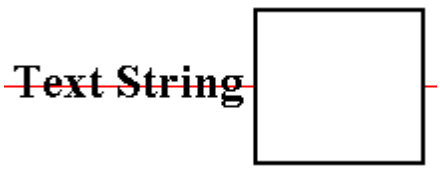
Units of measurement in TRichView [144]

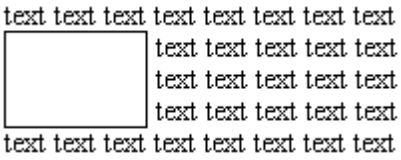
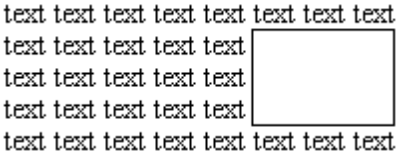
10.72 TRVVAlign

Unit [VCL/FMX] RVStyle / fmxRVStype;

```
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	

Align	Scheme
rvvaLeft	
rvvaRight	

Note: non-standard values of HTML align attributes (abstop, absbottom, absmiddle) are used for export of rvvaAbs*** values.

This property can be applied to:
- pictures [169] and *hot-pictures* [172]
- *bullets* [176] and *hotspots* [178]
- controls [174]
- label items [181]
- numbered sequences [183]

See also methods of TCustomRichView [216]:
- GetItemVAlign [311],
- SetItemVAlign [369].

See also properties of TCustomRichView [216]:
- ClearLeft [232],
- ClearRight [232].

See also methods of TCustomRichViewEdit [469]:
- GetCurrentItemVAlign [520],
- SetItemVAlignEd [575],
- SetCurrentItemVAlign [575],
- ClearTextFlow [509].

See also properties of TRVStyle [739]:
- FloatingLineColor [747]

10.73 TRVYesNoAuto

Unit [VCL/FMX] RichView / fmxRichView;

Yes/No/Auto values

Syntax:

```
type
  TRVYesNoAuto = (rvynaNo, rvynaYes, rvynaAuto);
```

10.74 TRVZoomPercent

Unit [VCL/FMX]: RVType / fmxRVType.

VCL and LCL:

```
type
  TRVZoomPercent = Integer;
```

FireMonkey:

```
type
  TRVZoomPercent = Single;
```

11 Global constants and variables

Global Variables and Typed Constants

Editing, Caret

- RichViewEditCaretWidthMode, RichViewEditCaretWidth[1108] define the caret width;
- RichViewEditCaretHeightExtra[1108]: Integer, allows making the caret shorter or longer;
- RichViewEditMinCaretHeight, RichViewEditMaxCaretHeight[1108]: Integer – minimum/maximum possible caret height;
- RichViewEditCustomCaretSize[1108]: Boolean allows changing caret size in an event.
- RichViewEditCaretPosition[1108]: TRVHorizontalCaretPosition defines how the caret is aligned horizontally relative to the insertion point.

Editing, Other

- RichViewEditDefaultProportionalResize[1113]: Boolean, specifies whether images are resized proportionally with the mouse;
- RichViewEditEnterAllowsEmptyMarkeredLines[1113]: Boolean, enables or disables autodeletion of list marker from new empty lines;
- RichViewUnicodeInput[1118]: TRichViewUnicodeInput, specifies how types Unicode characters are processed (in non-Unicode versions of Delphi);
- RichViewTableAutoAddRow[1117]: Boolean, allows or disallows adding new table rows using Tab key, as well as deletion of table rows and columns using Backspace key.
- RichViewCtrlUpDownScroll[1112]: Boolean, specifies how Ctrl + Up and Ctr + Down keys are processed.

Saving and Loading

- RVDefaultLoadProperties[1119]: TRVDefaultLoadProperties[1026], default properties for text and paragraph loaded from Markdown, HTML, RTF, DocX.
- RichViewSavePInHTML[1107]: Boolean, instructs always using <p> for saving in HTML;
- RichViewSaveDivInHTMLEx[1107]: Boolean, instructs always using for saving in HTML;
- RichViewCSSUseSystemColors[1112]: Boolean enables/disables saving system colors in CSS;
- RichViewSavePageBreaksInText[1107]: Boolean, allows or disallows saving page breaking characters in plain text;
- RichViewDoNotCheckRVFStyleRefs[1112]: Boolean, allows turning off checking for invalid style indices when loading RVF;

- RichViewResetStandardFlag[1115]: Boolean, specifies whether to reset Standard[804] property to False when inserting RVF;
- RichViewPixelsPerInch[1115]: Integer, allows overriding the screen resolution;
- RichViewAutoAssignNormalStyleTemplate[1111]: Boolean, specifies whether to assign "Normal" style template when loading unstyled paragraphs from RVF and RTF.
- RichViewTableDefaultRTFAutofit[1117]: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.
- RichViewSaveHyperlinksInDocXAsFields[1116] defines how hyperlinks are saved in DocX.
- RichViewRemoveNbSpAfterImport[1115] instructs to removed non-breaking spaces from empty lines after importing HTML and Markdown.
- DefaultRVFPixelsPerInch[1109] defines "pixels per inch" value for old RVF files.

Tables

- RichViewTableAutoAddRow[1117]: Boolean, allows or disallows adding new table rows using Tab key, and deleting rows/columns using Backspace key;
- RichViewAllowCopyTableCells[1110]: Boolean allows preventing copying multicell selection in a table to the Clipboard.
- RichViewTableDefaultRTFAutofit[1117]: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.

Formatting and Displaying

- RichViewJustifyBeforeLineBreak[1114]:Boolean, specifies whether lines before line breaks are justified (expanded);
- RichViewStringFormatMethod[1116]:Boolean, allows activating an alternative text measuring method;
- RichViewShowGhostSpaces[1116]: Boolean, allows showing space characters "hidden" between lines.
- RVVisibleSpecialCharacters[1129]: TRVSpecialCharacters specifies which special characters can be shown.
- RVAlwaysShowPlaceholders[1119]: Boolean, allows showing placeholders for text boxes even if special characters are not shown.
- RVThumbnailMaker[1129]: TRVThumbnailMaker[1041], provides access to an object used to create scaled images;
- RichViewMaxPictureCount[1114]: Integer – maximal count of "active" images in TRichView control.
- RichViewMaxThumbnailCount[1114]: Integer – maximal count of scaled images stored in TRichView control.

Printing

- RichViewPixelsPerInch[1115]: Integer, allows overriding the screen resolution;
- RichViewAlternativePicPrint[1110]: Boolean, allows using an alternative picture printing method.

Live spelling check

- RichViewApostropheInWord[1111]: Boolean, specifies how apostrophe characters are processed when finding word boundaries.

Singleton classes

- RVControlsPainter[1119]: TRVControlsPainter[1019] draws standard controls (buttons, scrollbars, list boxes, etc.).
- RVDefaultLoadProperties[1119]: TRVDefaultLoadProperties[1026], defines default properties for text and paragraph loaded from HTML, Markdown, RTF, DocX files.
- RVGraphicHandler[1125]: TRVGraphicHandler[1033], provides access to an object used to manage graphic classes.
- RVThumbnailMaker[1129]: TRVThumbnailMaker[1041], provides access to an object used to create scaled images;

Other

- RichViewCompareStyleNames[1111]: Boolean, specifies whether StyleName[805] property is used when comparing (and searching for) text, paragraph or list styles.

Global Functional Variables

- RVIsCustomURL[1126]: TCustomRVIsURLFunction, allows defining your own URL prefixes.
- RVGetSystemColor[1125]: TRVGetSystemColorFunction, allows using custom color schemes.

Constants

- **rvs*** ** constants[1127]: item types;
- **rveipc*** ** constants[1119]: integer properties of items;
- **rvespc*** **[1124] constants[1119]: string properties of items;
- Constants and variables related to Clipboard[1104].
- Color constants[1105]
- Pen style constants[1110]

11.1 Clipboard Related Constants and Variables

Unit [VCL/FMX]: RVStr / fmxRVStr;

Clipboard Format Names

Names of the Clipboard formats for copying RVF[129], RTF, URL, HTML.

Syntax

```
const
  RVFORMATNAME    = 'RichView Format';
  RTFORMATNAME    = 'Rich Text Format';
  URLFORMATNAME   = 'UniformResourceLocator';
  URLFORMATNAMEW   = 'UniformResourceLocatorW';
  HTMLFORMATNAME = 'HTML Format';
```

Clipboard Format Identifiers

```
var
  CFRV_RVF, CFRV_RTF, CFRV_HTML, CFRV_URL, CFRV_URLW: Cardinal;
  ...
  CFRV_RVF  := RegisterClipboardFormat(RVFORMATNAME);
  CFRV_RTF  := RegisterClipboardFormat(RTFORMATNAME);
  CFRV_URL  := RegisterClipboardFormat(URLFORMATNAME);
  CFRV_URLW := RegisterClipboardFormat(URLFORMATNAMEW);
  CFRV_HTML := RegisterClipboardFormat(HTMLFORMATNAME);
```

11.2 Color constants

Unit [VCL/FMX]: RVTypes / fmxRVTypes;

These constants define colors.

Syntax (VCL and LCL)

```
const
  rvclNone          = clNone;
  rvclWhite         = clWhite;
  rvclBlack         = clBlack;
  rvclRed           = clRed;
  rvclGreen         = clGreen;
  rvclLime          = clLime;
  rvclBlue          = clBlue;
  rvclSilver        = clSilver;
  rvclGray          = clGray;
  rvclAqua          = clAqua;
  rvclFuchsia       = clFuchsia;
  rvclYellow        = clYellow;
  rvclNavy          = clNavy;
  rvclTeal          = clTeal;
  rvclPurple        = clPurple;
  rvclMaroon        = clMaroon;
  rvclOlive         = clOlive;
  rvclInfoBk        = clInfoBk;
  rvclInfoText      = clInfoText;
  rvclWindow        = clWindow;
  rvclWindowText    = clWindowText;
  rvclHighlight     = clHighlight;
  rvclHighlightText = clHighlightText;
  rvclScrollbar     = clScrollbar;
  rvclActiveBorder  = clBtnShadow;
  rvclActiveCaption = clActiveCaption;
  rvclAppWorkSpace  = clAppWorkSpace;
  rvclBackground    = clBackground;
  rvclBtnText       = clBtnText;
  rvclCaptionText   = clCaptionText;
  rvclGrayText      = clGrayText;
  rvclInactiveBorder = clInactiveBorder;
  rvclInactiveCaption = clInactiveCaption;
  rvclInactiveCaptionText = clInactiveCaptionText;
  rvclGradientActiveCaption = clGradientActiveCaption;
  rvclHotLight      = clHotLight;
  rvclMenu          = clMenu;
  rvclMenuText      = clMenuText;
  rvcl3DDkShadow    = cl3DDkShadow;
  rvclBtnFace       = clBtnFace;
  rvcl3DLight       = cl3DLight;
  rvclBtnHighlight  = clBtnHighlight;
  rvclBtnShadow     = clBtnShadow;
  rvclWindowFrame   = clWindowFrame;
```

Syntax (FireMonkey)

```
const
  rvclNone          = TAlphaColorRec.Null;
  rvclWhite         = TAlphaColorRec.White;
  rvclBlack         = TAlphaColorRec.Black;
  rvclRed           = TAlphaColorRec.Red;
  rvclGreen         = TAlphaColorRec.Green;
  rvclLime          = TAlphaColorRec.Lime;
```

```
  rvclBlue          = TAlphaColorRec.Blue;
  rvclSilver        = TAlphaColorRec.Silver;
  rvclGray          = TAlphaColorRec.Gray;
  rvclAqua          = TAlphaColorRec.Aqua;
  rvclFuchsia       = TAlphaColorRec.Fuchsia;
  rvclYellow        = TAlphaColorRec.Yellow;
  rvclNavy          = TAlphaColorRec.Navy;
  rvclTeal          = TAlphaColorRec.Teal;
  rvclPurple        = TAlphaColorRec.Purple;
  rvclMaroon        = TAlphaColorRec.Maroon;
  rvclOlive         = TAlphaColorRec.Olive;
  rvclWindow        = rvclWhite;
  rvclWindowText    = rvclBlack;
  rvclInfoBk        = TAlphaColorRec.Lightyellow;
  rvclInfoText      = rvclWindowText;
  rvclHighlight     = $FF0078FF;
  rvclHighlightText = rvclWhite;
  rvclScrollbar     = TAlphaColorRec.Lightgray;
  rvclActiveBorder  = TAlphaColorRec.Lightgray;
  rvclActiveCaption = TAlphaColorRec.SkyBlue;
  rvclAppWorkSpace  = TAlphaColorRec.Darkgray;
  rvclBackground    = TAlphaColorRec.Black;
  rvclBtnText       = rvclWindowText;
  rvclCaptionText   = rvclWindowText;
  rvclGrayText      = TAlphaColorRec.Dimgray;
  rvclInactiveBorder = TAlphaColorRec.White;
  rvclInactiveCaption = TAlphaColorRec.Lightsteelblue;
  rvclInactiveCaptionText = rvclWindowText;
  rvclGradientActiveCaption = TAlphaColorRec.LightBlue;
  rvclGradientInactiveCaption = TAlphaColorRec.PowderBlue;
  rvclHotLight      = TAlphaColorRec.DodgerBlue;
  rvclMenu          = TAlphaColorRec.Gainsboro;
  rvclMenuBar       = rvclMenu;
  rvclMenuText      = rvclWindowText;
  rvclMenuHighlight = rvclHighlight;
  rvcl3DDkShadow    = TAlphaColorRec.Dimgray;
  rvclBtnFace       = TAlphaColorRec.Lightgray;
  rvcl3DLight       = TAlphaColorRec.Gainsboro;
  rvclBtnHighlight  = TAlphaColorRec.Whitesmoke;
  rvclBtnShadow     = TAlphaColorRec.Darkgray;
  rvclWindowFrame   = TAlphaColorRec.Dimgray;
```

See also:

- TRVColor[1063] type

11.3 Constants Related to HTML and Text

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
var
  RichViewSavePInHTML: Boolean = False;
  RichViewSaveDivInHTMLEx: Boolean = False;
```

```
  RichViewSavePageBreaksInText: Boolean = False;
```

Set **RichViewSavePInHTML** to *True* is you want SaveHTML [342] to write <p> instead of in HTML when HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstSimplified*. This variable does not affect saving when HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstNormal*. This setting changes appearance of resulting HTML documents (usually browsers add additional spacing between <p>).

Set **RichViewSaveDivInHTMLEx** to *True* is you want SaveHTML [342] to write instead of <p> in HTML when HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstNormal*. This variable does not affect saving when HTMLSaveProperties [244].HTMLSavingType [441] = *rvhtmlstSimplified*. 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.

11.4 Constants Related to the Caret

These typed constants allow to control the caret appearance in RichViewEdit [469].

Unit [VCL/FMX]: RVERVData / fmxRVERVData;

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 588 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 1058, 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.

11.5 DefaultRVFPixelsPerInch

Unit [VCL/FMX]: RVItem / fmxRVItem;

Syntax

```
const
  DefaultRVFPixelsPerInch: Integer = 0;
```

This value is used as PPI (pixels per inch) for RVF files saved by TRichView versions prior to 17.6.

Since version 17.6, TRichView saves PPI value in RVF file. This value is used to scale controls from RVF's PPI to the actual PPI. For older RVF, the value of this constant is used.

0 is a special value meaning no scaling.

11.6 Pen style constants

Unit [VCL/FMX]: RVTypes / fmxRVTypes;

These constants define pen styles.

Syntax (VCL and LCL)

```
const
  rvpsSolid         = psSolid;
  rvpsDash          = psDash;
  rvpsDot           = psDot;
  rvpsDashDot       = psDashDot;
  rvpsDashDotDot    = psDashDotDot;
  rvpsClear         = psClear;
  rvpsInsideFrame   = psInsideFrame;
```

Syntax (FireMonkey)

```
type
  TRVPenStyle = (rvpsSolid, rvpsDash,
    rvpsDot,  rvpsDashDot,
    rvpsDashDotDot, rvpsClear,
    rvpsInsideFrame);
```

See also:

- TRVPenStyle [1084]

11.7 RichViewAllowCopyTableCells

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
var
  RichViewAllowCopyTableCells: Boolean = True;
```

Set to *False* to disallow copying multicell selection (it will be copied as empty RVF [129]).

11.8 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.

11.9 RichViewApostropheInWord

Unit [VCL/FMX]: RVThread / fmxRVThread;

Syntax

```
var
  RichViewApostropheInWord: Boolean = True;
```

Specifies how apostrophe characters are processed in live spelling.

These characters are: ' (#$0027) and ’ (#$2019).

If *True*: apostrophes are allowed in middle of words, even if they are included in Delimiters [235] 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 [235] and assign RichViewApostropheInWord=*False*.

Note: apostrophe is not included in Delimiters [235] by default.

11.10 RichViewAutoAssignNormalStyleTemplate

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
const
  RichViewAutoAssignNormalStyleTemplate: Boolean = True;
```

(introduced in version 14)

This constant controls how unstyled paragraphs (i.e. paragraphs not linked [805] to any style template [843]) are loaded from RTF and RVF files/streams, if style templates are used [263].

If *True*, "Normal" style template [799] (if it exists) is assigned to paragraph styles [837] 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 [260] property.

See also:

- Styles and style templates [103].

11.11 RichViewCompareStyleNames

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
const
  RichViewCompareStyleNames: Boolean = False;
```

If set to *True*, methods for searching and comparing styles take StyleName [805] property into account.

11.12 RichViewCSSUseSystemColors

Unit: RVHtmlSave;

Syntax

```
const
  RichViewCSSUseSystemColors: Boolean = True;
```

Enables using system colors when saving HTML with CSS.

Color	CSS value
clWindow	canvas
clWindowText	canvastext
clBtnFace	buttonface
clBtnText	buttontext
clGrayText	graytext
clHighlight	highlight
clHighlightText	hightlightext

Older browsers do not support these colors in CSS. You can disable saving system colors by assigning **RichViewCSSUseSystemColors** = False. *clWindowText* will be saved as black, *clWindow* as white, other system colors will be converted to RGB depending in the system settings.

11.13 RichViewCtrlUpDownScroll

Unit [VCL/FMX]: RVEdit / fmxRVEdit;

Syntax

```
const
  RichViewCtrlUpDownScroll: Boolean = True;
```

(introduced in version 20)

This constant specifies how **Ctrl***+**Up** and **Ctrl***+**Down** keys are processed in TRichViewEdit[469].

True: scroll the editor (like in Embarcadero RAD Studio code editor)

False: move the caret to the beginning of the current/next paragraph (like in Microsoft Word).

* on macOS, **Command** is used instead of **Ctrl**.

11.14 RichViewDoNotCheckRVFStyleRefs

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
const
```

```
  RichViewDoNotCheckRVFStyleRefs: Boolean = False;
```

(introduced in version 10)

If set to *True*, TRichView ignore invalid style references when loading RVF documents [129].

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 [1031].

11.15 RichViewDoNotMergeNumbering

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
const
  RichViewDoNotMergeNumbering: Boolean = False;
```

RichViewDoNotMergeNumbering affects inserting RVF [129]. 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.

11.16 RichViewEditDefaultProportionalResize

Unit [VCL/FMX]: RVERVData / fmxRVERVData;

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)

11.17 RichViewEditEnterAllowsEmptyMarkeredLines

Unit [VCL/FMX]: RVERVData / fmxRVERVData;

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**.

11.18 RichViewJustifyBeforeLineBreak

Unit [VCL/FMX]: CRVFData / fmxCRVFData;

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.

11.19 RichViewMaxPictureCount

Unit [VCL/FMX]: RvGrCache / fmxRvGrCache;

Syntax

```
var
  RichViewMaxPictureCount: Integer  = 100;
```

(introduced in version 16)

Specifies the maximal count of images stored in TCustomRichView[216] in an "active" form, i.e. as TRVGraphic[1032] objects. Other pictures are "deactivated", i.e. stored in a memory stream until they are requested. This activation/deactivation allows saving system resources and memory.

Changes of value of this property do not affect TRichView components created before the change.

See also:

- Smooth image scaling[125]
- RichViewMaxThumbnailCount[1114]

11.20 RichViewMaxThumbnailCount

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
var
  RichViewMaxThumbnailCount = 50;
```

(introduced in version 15)

Specifies the maximal count of scaled images stored in TCustomRichView[216].

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 [125]
- RichViewMaxPictureCount [1114]

11.21 RichViewPixelsPerInch

Number of pixels in one screen inch.

Unit [VCL/FMX]: RVFuncs / fmxRVFuncs;

Syntax

```
const
  RichViewPixelsPerInch : Integer = 0;
```

If 0 (default), TRichView uses the screen resolution (Screen.PixelsPerInch), or, if the application supports it, resolution of its monitor (in Delphi 10.1+).

If non zero, this value is used.

If you use TRVRuler component (from RichViewActions) and if you assign a positive value to RichViewPixelsPerInch, assign the same value to RVRuler1.ScreenRes property,

DEPRECATED!

This variable was useful in TRichView versions prior to 18 to provide printing, import and export independent of the screen resolution. In the new version of TRichView, everything is independent of the screen resolution by default.

Instead of this constant, the following properties must be used:

- TRVStyle [739].UnitsPixelsPerInch [765]
- TRichView [216].DocumentPixelsPerInch [239]

11.22 RichViewRemoveNbSpAfterImport

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
var
  RichViewRemoveNbSpAfterImport: Boolean = True;
```

If *True*, non-breaking spaces are removed from empty lines after importing HTML and Markdown. Non-breaking spaces are often used in these formats to create empty lines.

11.23 RichViewResetStandardFlag

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
const
  RichViewResetStandardFlag: Boolean = True;
```

If *True*, Standard [804] properties of styles which were added from inserted RVF [129] will be reset to *False* (it only affects RVF insertion, not loading).

If this flag is *True*, new styles having Standard[804]=*True* may appear only after calling LoadRVF[336] or LoadRVFFromStream[337].

11.24 RichViewSaveHyperlinksInDocXAsFields

Unit [VCL/FMX]: CRVData / fmxCRVData;

Syntax

```
const
    RichViewSaveHyperlinksInDocXAsFields = False;
```

This constant switches between two methods of hyperlinks saving in DocX files.

If *False*:

SaveDocX[341] 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.

11.25 RichViewShowGhostSpaces

Unit [VCL/FMX]: CRVData / fmxCRVData;

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[247]. Caret cannot be placed after these spaces.

11.26 RichViewStringFormatMethod

Unit [VCL/FMX]: CRVData / fmxCRVData;

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[763]=*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.

Otherewise, the components always choose the TRichView method.

11.27 RichViewTableAutoAddRow

Unit [VCL/FMX]: RVTable / fmxRVTable;

Syntax

```
const
  RichViewTableAutoAddRow: Boolean = True;
```

(introduced in version 10)

Set to *False* to disable:

1. auto insertion of new table row when the user presses **Tab** key in the last table[179] cell;
2. deletion of table, or its rows, or its columns when the user presses **Backspace** key.

The row auto insertion works only in TRichViewEdit[469] and TDBRichViewEdit[616].

11.28 RichViewTableDefaultRTFAutofit

Unit [VCL/FMX]: RVTable / fmxRVTable;

Syntax

```
const
  RichViewTableDefaultRTFAutofit: Boolean = False;
```

Assigning *True* to this variable works like inclusion of *rvtoRTFAllowAutofit* in Options[916] properties of all tables[179].

11.29 RichViewUnicodeInput

Unit [VCL/FMX]: RVEdit / fmxRVEdit;

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 [135] 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.

11.30 RV_CP_DEFAULT

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

```
const
  RV_CP_DEFAULT = $FFFFFFFF;
```

Default code page value.

This value is supported only in the following places:

- for CodePage parameter of LoadFromFile, LoadFromFileEx [328] and LoadFromStream, LoadFromStreamEx [329];
- TRichView [216].Document [238].CodePage [431]
- TDBRichView [604].CodePage [609]
- TDBRichViewEdit [616].CodePage [622]

When this value is specified, the component uses the following code pages:

- CP_UTF8 (UTF-8) for Markdown
- CP_ACP (code page of the default Windows language) for plain text files in Windows
- CP_UTF8 for plain text files in other (non-Windows) platforms.

11.31 RVAlwaysShowPlaceholders

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
const
  RVAlwaysShowPlaceholders: Boolean = False;
```

(introduced in version 19)

If *True*, placeholder icons are always shown in places of insertion of text boxes [188].

If *False*, these icons are shown only at the caret position, or if *rvoShowSpecialCharacters* is included in TCustomRichView.Options [247], and *rvscPlaceholders* is included in RVVisibleSpecialCharacters [1129].

11.32 RVControlsPainter

Unit [VCL/FMX]: RVControls / fmxRVControls;

Syntax

```
var RVControlsPainter: TRVControlsPainter[1019];
```

(introduced in version 18)

This variable returns a singleton object used to draw standard controls (buttons, scrollbars, list boxes, etc.).

Currently, this variable is not used in TRichView itself. However, it is used in ScaleRichView and (especially) in SRVControls.

11.33 RVDefaultLoadProperties

Unit [VCL/FMX]: RVDefReadProps / fmxRVDefReadProps;

Syntax

```
var
  RVDefaultLoadProperties: TRVDefaultLoadProperties[1026];
```

(introduced in version 20)

This variable returns a singleton object that is used by all TRichView controls. It defines default properties for text and paragraph loaded from HTML, Markdown, RTF, DocX.

11.34 rveipc*** constants

These constants identify integer properties of TRichView items [163].

They are used in the following methods:

- TRichView.GetExtraItemIntPropertyEx [307]
- TRichView.SetExtraItemIntPropertyEx [366]
- TRichViewEdit.SetItemExtraIntPropertyExEd [572]
- TRichViewEdit.GetCurrentItemExtraIntPropertyEx [518]
- TRichViewEdit.SetCurrentItemExtraIntPropertyEx [566]

Properties of Bullets[176] and Hotspots[178]

These properties are defined in RVItem unit.

These properties provides an alternative to the following methods:

- TRichView: SetBulletInfo[359], GetBulletInfo[297], GetHotspotInfo[303], SetHotspotInfo[364];
- TRichViewEdit: SetBulletInfoEd[558], SetCurrentBulletInfo[562], GetCurrentBulletInfo[513], SetHotspotInfoEd[570], SetCurrentHotspotInfo[565], GetCurrentHotspotInfo[515].

Constant	Value	Property
rveipcImageIndex	200	Image index (for *bullets* and *hotspots*)
rveipcHotImageIndex	201	Image index below the mouse pointer (for *hotspots*)

Properties of Controls[174]

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.
rveipcDPIScalable	201	If non-zero (default), controls are scaled from RVF's PPI (pixels per inch) to the actual PPI. See also: RichViewPixelsPerInch[1115]

Properties of Breaks[173]

These properties are defined in RVItem unit.

These properties provide an alternative to the following methods:

- TRichView: SetBreakInfo[358], GetBreakInfo[296];
- TRichViewEdit: SetBreakInfoEd[557], SetCurrentBreakInfo[561], GetCurrentBreakInfo[512].

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 [181]

These properties are defined in RVLabelItem unit.

They identify properties of label items [181] and items of all types inherited from them (numbered sequences [183], footnotes [185], endnotes [184], references to notes [190]).

Constant	Value	Property
rveipcProtectTextStyleNo	200	ProtectTextStyleNo [972] *
rveipcRemoveInternalLeading	201	RemoveInternalLeading [972] *
rveipcCursor	202	Cursor [971]
rveipcMinWidth	203	MinWidth [971]
rveipcAlignment	204	Alignment [971] *
rveipcRemoveSpaceBelow	205	RemoveSpaceBelow [972] *
rveipcUseTypoMetric	206	UseTypeMetric

* the property is type-casted to Integer

Properties of Numbered sequences [183]

These properties are defined in RVSeqItem unit.

They identify properties of numbered sequences [183] and items of all types inherited from them (footnotes [185], endnotes [184]).

Constant	Value	Property
rveipcSeqStartFrom	300	StartFrom [981]
rveipcSeqReset	301	Reset [981] *
rveipcSeqNumberType	302	NumberType [980] *,**

* the property is type-casted to Integer

** read-only for footnotes [185], endnotes [184], sidenotes [186] and text box items [188].

Properties of Sidenotes [186]

Position properties

These properties are defined in RVFloatingPos unit.

They identify sub-properties of BoxPosition [993] property of sidenotes [186] and text box items [188].

Constant	Value	Property
rveipcFloatHorizontalAnchor	500	HorizontalAnchor[998]*
rveipcFloatHorizontalPositionKind	501	HorizontalPositionKind[999]*
rveipcFloatHorizontalOffset	502	HorizontalOffset[999]
rveipcFloatHorizontalAlign	503	HorizontalAlignment[999]*
rveipcFloatVerticalAnchor	504	VerticalAnchor[1002]*
rveipcFloatVerticalPositionKind	505	VerticalPositionKind[1003]*
rveipcFloatVerticalOffset	506	VerticalOffset[1003]
rveipcFloatVerticalAlign	507	VerticalAlignment[1003]*
rveipcFloatRelativeToCell	508	RelativeToCell[1002]*
rveipcFloatPositionInText	509	PositionInText[1001]*

* the property is type-casted to Integer

Other properties

These properties are defined in RVFloatingBox unit.

They identify sub-properties of BoxProperties[992] property of sidenotes[186] and text box items[188].

Constant	Value	Property
rveipcBoxWidth	600	Width[996]
rveipcBoxHeight	601	Height[995]
rveipcBoxWidthType	602	WidthType[996]*
rveipcBoxHeightType	603	HeightType[995]*
rveipcBoxBorderColor	604	Border[995].Color[853]
rveipcBoxBorderWidth	605	Border[995].Width[854]
rveipcBoxBorderInternalWidth	606	Border[995].InternalWidth[853]
rveipcBoxBorderStyle	607	Border[995].Style[854]*
rveipcBoxBorderVisibleBorders_Left	608	Border[995].VisibleBorders[854].Left*
rveipcBoxBorderVisibleBorders_Top	609	Border[995].VisibleBorders[854].Top*
rveipcBoxBorderVisibleBorders_Right	610	Border[995].VisibleBorders[854].Right*

Constant	Value	Property
rveipcBoxBorderVisibleBorders_Bottom	611	Border[995].VisibleBorders[854].Bottom*
rveipcBoxBorderBorderOffsets_Left	612	Border[995].BorderOffsets[851].Left
rveipcBoxBorderBorderOffsets_Top	613	Border[995].BorderOffsets[851].Top
rveipcBoxBorderBorderOffsets_Right	614	Border[995].BorderOffsets[851].Right
rveipcBoxBorderBorderOffsets_Bottom	615	Border[995].BorderOffsets[851].Bottom
rveipcBoxColor	616	Background[995].Color[852]
rveipcBoxPadding_Left	617	Background[995].BorderOffsets[851].Left
rveipcBoxPadding_Top	618	Background[995].BorderOffsets[851].Top
rveipcBoxPadding_Right	619	Background[995].BorderOffsets[851].Right
rveipcBoxPadding_Bottom	620	Background[995].BorderOffsets[851].Bottom
rveipcBoxVAlign	621	VAlign[996]*
rveipcBoxOpacity	622	Background[995].Opacity[852]

* the property is type-casted to Integer

Properties of "Page number"[190] and "Page count"[192] fields

These properties are defined in RVFieldItems unit.

Constant	Value	Property
rveipcPageNumberType	600	NumberType[1009]*

* the property is type-casted to Integer

Properties of mathematical expressions[193]

These properties are defined in RVMathItem unit.

Constant	Value	Property
rveipcFontSizeDouble	200	FontSizeDouble[976]
rveipcTextColor	201	TextColor[977]

Constant	Value	Property
rveipcDisplayInline	202	DisplayInline[975]*

* the property is type-casted to Integer

See also

- rvespc*** constants[1124]

11.35 RVEMPTYTAG

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

```
const
  RVEMPTYTAG = '';
```

11.36 rvespc*** constants

These constants identify string properties of TRichView items[163].

They are used in the following methods:

- TRichView.GetExtraItemStrPropertyEx[308]
- TRichView.SetExtraItemStrPropertyEx[367]
- TRichViewEdit.SetItemExtraStrPropertyExEd[573]
- TRichViewEdit.GetCurrentItemExtraStrPropertyEx[519]
- TRichViewEdit.SetCurrentItemExtraStrPropertyEx[567]

Properties of Label items[181]

These properties are defined in RVLabelItem unit.

They identify properties of label items[181] and items of all types inherited from them (numbered sequences[183], footnotes[185], endnotes[184], sidenotes[186], references to notes[190], text boxes[188], page numbers[190], page counts[192]).

Constant	Value	Property
rvespcText	200	Text[972]*

* read-only for numbered sequences[183], footnotes[185], endnotes[184], sidenotes[186], references to notes[190]; ignored for text boxes[188]

Properties of Numbered sequences[183]

These properties are defined in RVSeqItem unit.

Constant	Value	Property
rvespcSeqName	300	SeqName[981]
rvespcSeqFormatString	301	FormatString[980]

Properties of mathematical expressions [193]

These properties are defined in RVMathItem unit.

Constant	Value	Property
rvespcText	200	Text [977]
rvespcFontName	201	FontName [976]

See also

- rveipc*** constants [1119]

11.37 RVGetSystemColor

Unit [VCL/FMX]: RVColorFuncs / fmxRVColorFuncs;

Returns value of

Syntax

```
type
  TRVGetSystemColorFunction =
    function (AControl: TControl; AColor: TRVColor[1063]): TRVColor;

const
  RVGetSystemColor: TRVGetSystemColorFunction = nil;
```

(introduced in version 23)

In VCL application, this function can be used to return values of system colors.

This function (if it is defined) is called for TRichView controls that have UseVCLThemes [264] = *True*.

Parameters:

AControl – the control where the painting is being performed.

AColor – the original color.

Return value:

The color that will be used instead of **AColor**.

11.38 RVGraphicHandler

Unit [VCL/FMX]: RVGrHandler / fmxRVGrHandler;

Syntax

```
var RVGraphicHandler: TRVGraphicHandler[1033];
```

(introduced in version 15)

This variable returns an object used by TRichView components to manage graphic classes.

11.39 RVIsCustomURL

Unit [VCL/FMX]: RVFuncs / fmxRVFuncs;

Syntax

```
type
  TCustomRVIsURLFunction = function (const Word: TRVUnicodeString[1099]): 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[1053] function.

See demos:

- Demos*\Assorted\Hypertext\URLs\

11.40 RVParagraphMarks

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
type
  TRVParagraphMarks = (rvpmStandard, rvpmArrows);
const
  RVParagraphMarks: TRVParagraphMarks = rvpmStandard;
```

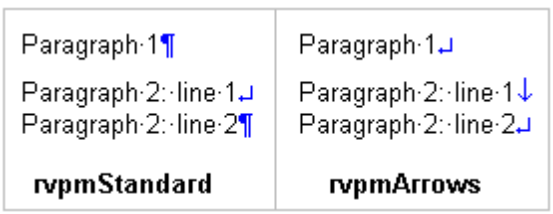
(introduced in version 13)

Defines a set of symbols used to display ends of paragraphs and lines.

These symbols are displayed if *rvoShowSpecialCharacters* is included in TCustomRichView.Options[247], and *rvscParagraph* is included in RVVisibleSpecialCharacters[1129] 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[762]

11.41 rvs*** constants

There are three groups of **rvs*** ** constants:

- identifiers of types of RichView[216] items[90];
- indexes in the collection of text styles (RVStyle.TextStyles[764]) corresponding to the initial entries;
- values used by TRichView internally when saving and loading RVF[129] files (see RVF specification[148]).

Identifiers of Item Types

Use the first group of constants to compare with values returned by RichView.GetItemStyle[310] or TRichViewEdit.CurItemStyle[481].

Constant	Value	Item Type	Unit
rvsBreak	-1	break[173] (horizontal line)	RVStyle
rvsPicture	-3	picture[169]	RVStyle
rvsHotspot	-4	hotspot[178] (picture from ImageList - hyperlink)	RVStyle
rvsComponent	-5	inserted Delphi control[174]	RVStyle
rvsBullet	-6	bullet[176] (picture from ImageList)	RVStyle
rvsHotPicture	-10	hot picture[172] (picture - hyperlink)	RVStyle
rvsListMarker	-11	list marker[180] (bullet or numbering of paragraphs)	RVStyle
rvsTab	-12	tabulator[168]	RVStyle
rvsTable	-60	table[196]	RVTable
rvsLabel	-200	label[181] (non-text item looking like text)	RVLabelItem
rvsSequence	-202	numbered sequence[183]	RVSeqItem
rvsFootnote	-203	footnote[185]	RVNote
rvsEndnote	-204	endnote[184]	RVNote
rvsSidenote	-206	sidenotes[186]	RVSidenote
rvsTextBox	-207	text boxes[188]	RVSidenote
rvsNoteReference	-205	reference to the parent note[190]	RVNote
rvsPageNumber	-208	"page number" field[190]	RVFieldItems

Constant	Value	Item Type	Unit
rvsPageCount	-209	"page count" field [192]	RVFieldItems
rvsMathEquation	-210	mathematical expression [193]	RVMathItem
rvsChatFile	-212	file attached to a message [195]	RVChat

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 [90].

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 Use
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 [839] will be used. This feature is deprecated, use StyleTemplates [103] instead.

This value also has a special meaning in TRichView.OnReadField [393] event.

11.42 RVThumbnailMaker

Unit [VCL/FMX]: RVThumbMaker / fmxRVThumbMaker;

Syntax

```
var RVThumbnailMaker: TRVThumbnailMaker[1041];
```

(introduced in version 15)

This variable returns a singleton 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 [125]

11.43 RVVisibleSpecialCharacters

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

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 [247].

Value	Meaning
rvscSpace	If included, space characters are shown (as middle dots). Hidden spaces at the ends of lines are not shown, see

Value	Meaning
	RichViewShowGhostSpaces[1116]
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[831] (if defined); ▪ rectangle if any non-default printing option[833] is defined (never displayed for headings 1...9) ▪ heading level (1...9) ▪ text flow clearing[232] (for the next paragraph or this line break)
rvscPlaceholders	If included, special marks are painted at places of insertion of text box items[188]. See also RVAlwaysShowPlaceholders[1119].
rvcFloatingLines	If included, floating lines are displayed for left-and right-aligned items[1100], see TRVStyle.FloatingLineColor[747]

See also:

- RVParagraphMarks[1126]
- TRVStyle.SpecialCharactersColor[762]

12 How to...

How to...

(printing)

- Change page size and orientation for printing RichView[1131]
- Draw page numbers, headers and footers[1131]

(editing)

- Implement commands like "make bold", "apply font", etc.[1131]
- Implement Find and Replace dialogs[1132]
- Switch insert/overtype mode[1132]
- Move the caret to the beginning or to the end of document in editor[1133]
- Make Unicode editor[1134]
- Make a plain text editor[1134]
- Implement smart indents[1136]

(miscellaneous)

- Use third-party graphic classes with RichView[1136]
- Combine several documents in one[1136]
- Remove all formatting[1139]

12.1 change page size and orientation

TRVPrint[696] 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[711] method.

12.2 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[696] component by the methods SetHeader[716] and SetFooter[715].

See the example:

- Demos*\Assorted\Printing\Headers\

You can assign special headers and footers for the first page, odd/even pages.

You can insert page numbers[190] and page counts[192].

Drawing headers and footers yourself

If you need different headers for different pages, you need to draw them yourself, using OnPagePrepaint[719] event.

This demo shows how to draw a simple text as a header:

- Demos*\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[721] component for this.

Examples are shown in this forum topic: ***https://www.trichview.com/forums/viewtopic.php?f=3&t=327***

12.3 implement commands like "make bold", "apply font", etc.

The key method is ApplyStyleConversion[501].

Example:

- Demos*\Editors\Editor 2\

Alternatively, you can use ***RichViewActions***.

12.4 implement Find and Replace dialogs

Find

The key method is TRichView.SearchText[354] or TRichViewEdit.SearchText[553].

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[1049] or GetRVESearchOptions[1049].

Replace

Use TRichViewEdit.SearchText[553] to select the old string.

Then use InsertText[540] 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*\Editors\Editor 1\

Both Find and Replace dialogs are implemented in the demo:

- Demos*\Assorted\Search and Replace\

Alternatively, you can use ***RichViewActions***.

12.5 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[356];
end;

procedure TForm1.RichViewEdit1KeyPress(Sender: TObject; var Key: Char);
var rve: TCustomRichViewEdit[469];
    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[489];
  if rve.SelectionExists[356] then
    exit;
  ItemNo := rve.CurItemNo[480];
  Offs  := rve.OffsetInCurItem[488];
  if (Offs>=rve.GetOffsAfterItem[316](ItemNo)) then
  begin
    if (ItemNo+1<rve.ItemCount[244]) and
       not rve.IsFromNewLine[325](ItemNo+1) then
    begin
      inc(ItemNo);
      Offs := rve.GetOffsBeforeItem[317](ItemNo);
    end
    else
      exit;
  end;
  rve.SetSelectionBounds[372](ItemNo, Offs, ItemNo, Offs+1);
  rve.Invalidate;
end;
```

More details and demo: ***https://www.trichview.com/forums/viewtopic.php?t=171***

12.6 move the caret to the beginning or to the end of document in editor

Option 1

Moving to the beginning:

```
MyRichViewEdit.MoveCaret[542](rvcmTop);
```

Moving to the end:

```
MyRichViewEdit.MoveCaret[542](rvcmBottom);
```

Option 2

The caret is always at the end of selection.

Selection can be changed using method TRichViewEdit.SetSelectionBounds[372].

Moving to the beginning:

```
var
  ItemNo, Offs: Integer;
...
ItemNo := 0;
Offs   := MyRichViewEdit.GetOffsBeforeItem[317](ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo,Offs,ItemNo,Offs);
```

Moving to the end:

```
var
  ItemNo, Offs: Integer;
...
ItemNo := MyRichViewEdit.ItemCount[244]-1;
Offs   := MyRichViewEdit.GetOffsAfterItem[316](ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo,Offs,ItemNo,Offs);
```

See also:

- TRichViewEdit.CurItemNo[480] and OffsetInCurItem[488];
- Example how to move caret to the beginning of paragraph[602].

12.7 make Unicode editor

See in the topic Unicode in TRichView[135].

(This topic is obsolete. All text in TRichView is Unicode for all versions of Delphi)

12.8 make a plain text editor

How to make a plain text editor based on TRichViewEdit[469]

- Exclude all but text from AcceptDragDropFormat[478] property.
- Process OnPaste[594] 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) the
  begin
    s := Clipboard.AsText;
    Sender.InsertText[540](s, False);
  end;
  DoDefault := False;
end;
```

How to make ONE LINE plain text editor

- Exclude all but text from AcceptDragDropFormat[478] property.
- Process OnPaste[594] event (see below).
- Process OnOleDrop[591] event (see below).
- Disable all UI commands changing text and paragraph attributes or inserting multiline text.
- Set WordWrap[265] property = *False*.
- Include *rvoDoNotWantReturns* in EditorOptions[483].
- Set VSmallStep[264] property = 1, before calling Format[295].
- If you do not want tabulators as well, set RVStyle.SpacesInTab[761] = 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[216];
  const DataObject: IDataObject; Shift: TShiftState; X, Y: Integer;
  PossibleDropEffects: TRVOleDropEffects[1081];
  var DropEffect: TRVOleDropEffect[1081]; 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[540](MakeOneLineString(s), False);
  end;
```

```
  ReleaseStgMedium(StgMedium);
end;

procedure TMyForm.MyRichViewEditPaste(Sender: TCustomRichViewEdit[469];
  var DoDefault: Boolean);
var s: String;
begin
  if Clipboard.HasFormat(CF_TEXT) then
  begin
    s := Clipboard.AsText;
    Sender.InsertText[540](MakeOneLineString(s), False);
  end;
  DoDefault := False;
end;
```

12.9 implement smart indenting

You can find the examples in the support forum:
https://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[831] and LeftIndent[832] of paragraph, if **Tab** / **Backspace** is pressed when the caret is at the beginning of paragraph. First, it changes FirstIndent[831]. Next, if FirstIndent[831] is already changed, it changes LeftIndent[832]. The code changes indents only if the caret is at the beginning of non-empty paragraph.

12.10 combine several TRichView documents in one

See in the topic TCustomRichView.InsertRVFFromStream[323].

12.11 use third-party graphic classes with TRichView

About FireMonkey

All the information below is about VCL and LCL.

In FireMonkey, graphic classes are inherited from TRVGraphicFM[1032].

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[278] or InsertPicture[534].

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.trichview.com/resources/thirdparty/gifimage.zip***) (useful for Delphi 2006 and older)
- TJvGifImage from JEDI's JVCL ***https://wiki.delphi-jedi.org/*** (useful for Delphi 2006 and older)
- Gustavo Huffenbacher Daud's TPngObject ***www.trichview.com/resources/thirdparty/pngimage.zip*** (useful for Delphi 2007 and older)

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[124], assign TCustomRichView.AnimationMode[228] = *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[124]

Controlling automatic insertion of graphics when importing RTF, DocX, HTML, and Markdown

Loading external image files

Sometimes images are inserted automatically (for example, when importing images referred from RTF, DocX, Markdown or HTML).

Question: What graphic classes does TRichView use for loading graphic formats (such as GIF or PNG)?

Answer: TRichView uses the graphic class which is registered for the given file extension.

First, TRichView tried to detect image format by the file content, and choose the appropriate graphic class for this format.

If the first part was not succeed, it uses associations between file extensions and graphic classes.

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);
```

Loading images embedded in RTF, DocX or HTML

When importing images contained inside RTF or DocX, TRichView uses graphic classes, specified as default graphic classes for graphic format. See TRVGraphicHandlder.SetDefaultGraphicClass[1033].

When importing images contained inside HTML (base64-encoded), TRichView detects graphic format by content.

Workaround for Delphi bug in constructors of graphic classes (Delphi 5)

If you use Delphi 5, use RVGraphicHandler[1125].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 to HTML

By default, all non-web images are converted to JPEGs. It may be undesirable for some other formats.

You can specify which graphic classes must not be converted.

Use RVGraphicHandler[1125].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.
- TPortableNetworkGraphic (for Lazarus[160]).

Specifying PNG class

Specify graphic class for PNG images using Use RVGraphicHandler[1125].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[1125].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.

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, Svg images.

TcxPngImage becomes a default Png class for TRichView, but TdxSmartImage does not become a default Jpeg class, unless you call RVGraphicHandler.RegisterJpegGraphic explicitly.

12.12 create a chat window

Starting from TRichView v24, you can use TRVChat [640] component.

You can find the complete demo of a chat window on the support forum:
https://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 [1136]), including animations [124]. For additional ideas about emoticons, see the demo in Demos*\Assorted\Graphics\Emoticons\.

URL detection use RVIsURL and RVIsEmail [1053] functions. You can use RVIsCustomURL [1126] to define your own types of links.

12.13 remove text formatting

The code below removes text formatting from the whole document.

It converts indices of all text styles [764] and paragraph styles [757] to 0s. Optionally, it can keep hyperlinks (they are converted to the hypertext style with the lowest index).

Optionally, it can remove list markers [180] as well.

This code changes text items [167], tabulators [168] and list markers. All other item types remains unchanged.

Note 1: if the document contains both Unicode [135] 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[1014];
  RemoveMarkers, KeepLinks: Boolean);
var
  i, HypertextStyleNo: Integer;
  {.....................................................}
  procedure DoRemoveFormatting(RVData: TCustomRVData[1014]);
  var
    i, r, c, StyleNo, StyleNoTo: Integer;
    PB: Boolean;
    table: TRVTableItemInfo[901];
  begin
    for i := RVData.ItemCount[244]-1 downto 0 do
    begin
      RVData.GetItem[304](i).ParaNo := 0;
      StyleNo := RVData.GetItemStyle[310](i);
      case StyleNo of
        rvsTable:
          begin
            table := TRVTableItemInfo[901](RVData.GetItem[304](i));
            for r := 0 to table.RowCount[920]-1 do
              for c := 0 to table.ColCount[914]-1 do
                if table.Cells[912][r,c]<>nil then
                  DoRemoveFormatting(table.Cells[912][r,c].GetRVData[1017]);
          end;
        rvsListMarker:
          if RemoveMarkers then begin
            PB := RVData.PageBreaksBeforeItems[250][i];
            RVData.DeleteItems[291](i, 1);
            if i<RVData.ItemCount[244] then
            begin
              RVData.GetItem[304](i).SameAsPrev := False;
              RVData.GetItem[304](i).PageBreakBefore := PB;
            end;
          end;
        rvsTab:
          if RVData.GetRVStyle.TextStyles[764][
            TRVTabItemInfo(RVData.GetItem[304](i)).TextStyleNo].Jump[823] and
            KeepLinks then
            TRVTabItemInfo(RVData.GetItem[304](i)).TextStyleNo :=
```

```
            HypertextStyleNo
          else
          begin
            TRVTabItemInfo(RVData.GetItem(304)(i)).TextStyleNo := 0;
            RVData.SetItemTag(367)(i, '');
          end;
        1..MaxInt:
          begin
            if KeepLinks and
              RVData.GetRVStyle.TextStyles(764)[StyleNo].Jump(823) then
              StyleNoTo := HypertextStyleNo
            else
            begin
              StyleNoTo := 0;
              RVData.SetItemTag(367)(i, '');
            end;
            RVData.GetItem(304)(i).StyleNo := StyleNoTo;
          end;
      end;
    end;
  end;
  {.....................................................}
begin
  HypertextStyleNo := 0;
  if KeepLinks then
    for i := 0 to RVData.GetRVStyle.TextStyles(764).Count-1 do
      if RVData.GetRVStyle.TextStyles(764)[i].Jump(823) then
      begin
        HypertextStyleNo := i;
        break;
      end;
  DoRemoveFormatting(RVData);
end;
```

How to use:

```
  RemoveFormatting(RichViewEdit1.RVData(254), True, True);
  // from RVNormalize.pas, from RichViewActions
  NormalizeRichView(RichViewEdit1.RVData(254));
  RichViewEdit1.DeleteUnusedStyles(293)(True, True, True);
  RichViewEdit1.ClearUndo(509);
  RichViewEdit1.Format(295);
```

More information: *https://www.trichview.com/forums/viewtopic.php?t=2392*

Index

- A -

AcceptDragDropFormats 478
TRichViewEdit 478

AcceptPasteFormats 478
TRichViewEdit 478

Active 721
TRVVirtualPrinterProperties 721

ActualCurTextStyleNo 480
TCustomRichViewEdit 480

Add 277, 787, 793, 797, 859
TFontInfos 787
TParaInfos 793
TRichView 277
TRVListInfos 797
TRVListLevelCollection 859

AddBreak 269
TRichView 269

AddBreakEx 269
TRichView 269

AddBreakExTag 269
TRichView 269

AddBreakTag 269
TRichView 269

AddBullet 270
TRichView 270

AddBulletEx 270
TRichView 270

AddBulletExTag 270
TRichView 270

AddCheckpoint 271
TRichView 271

AddCheckpointTag 271
TRichView 271

AddControl 272
TRichView 272

AddControlEx 272
TRichView 272

AddControlExTag 272
TRichView 272

AddFile 671
TRVChatMessageExtraFiles 671

AddFmt 273
TRichView 273

AddFont 787
TFontInfos 787

AddFontEx 788
TFontInfos 788

AddFrom 855
TRVTabInfos 855

AddHotPicture 274
TRichView 274

AddHotPictureTag 274
TRichView 274

AddHotspot 275
TRichView 275

AddHotspotEx 275
TRichView 275

AddHotspotExTag 275
TRichView 275

AddItem 276
TRichView 276

AddMessage 645
TRVChat 645

AddNamedCheckpoint 271
TRichView 271

AddNamedCheckpointEx 271
TRichView 271

AddNamedCheckpointExTag 271
TRichView 271

AddNL 277
TRichView 277

AddNLATag 277
TRichView 277

AddNLTag 277
TRichView 277

AddNLWTag 277
TRichView 277

AddPicture 278
TRichView 278

AddPictureEx 278
TRichView 278

AddPictureExTag 278
TRichView 278

AddSpellingMenuItems 497
TRichViewEdit 497

AddStyles 646
TRVChat 646

AddTab 280
TRichView 280

AddTag 277
TRichView 277

AddTextNL 281
TRichView 281

AddTextNLA 281

AddTextNLA 281
TRichView 281
AddTextNLW 281
TRichView 281
AddTextStyle 767
TRVStyle 767
AddToAutoreplaceList 733
TRVSpellChecker 733
AddToDictionary 733
TRVSpellChecker 733
AddToIgnoreList 733
TRVSpellChecker 733
AddXXX methods of TRichView 107
AdjustControlPlacement 497
TRichViewEdit 497
AdjustControlPlacement2 497
TRichViewEdit 497
Align 856
TRVTabInfo 856
Alignment 829, 971
TCustomRVParaInfo 829
TRVLabelItemInfo 971
AllNumbered 842
TRVListInfo 842
AllowMarkdown 430, 608, 621
TDBRichView 608
TDBRichViewEdit 621
TRVDocumentProperty 430
AllowSelection 228
TRichView 228
AllowTransparentDrawing 723
TRVReportHelper 723
Animation in TRichView 124
AnimationMode 228
TRichView 228
AppendFrom 282
TRichView 282
AppendRVFFromStream 283
TRichView 283
ApplyListStyle 498
TRichViewEdit 498
ApplyParaStyle 498
TRichViewEdit 498
ApplyParaStyleConversion 499
TRichViewEdit 499
ApplyParaStyleTemplate 500
TRichViewEdit 500
ApplyStyleConversion 501
TRichViewEdit 501
ApplyStyleTemplate 501
TRichViewEdit 501
ApplyTextStyle 502
TRichViewEdit 502
ApplyTextStyleTemplate 502
TRichViewEdit 502
ApplyToParaStyle 848
TRVStyleTemplate 848
ApplyToTextStyle 849
TRVStyleTemplate 849
ArrowHeight 657
TRVChatCalloutProperties 657
ArrowWidth 657
TRVChatCalloutProperties 657
Assign 820, 825, 836, 840
TCustomRVFontInfo 820
TCustomRVParaInfo 836
TFontInfo 825
TParaInfo 840
AssignDefaultFileTypes 672
TRVChatMessageExtraFiles 672
AssignDocParameters 708
TRVPrint 708
AssignProperties 925
TRVTableItemInfo 925
AssignSoftPageBreaks 284
TCustomRichView 284
AssignSource 709
TRVPrint 709
AssignStyleTemplates 799
TRVStyleTemplateCollection 799
AssignTo 788, 793, 797, 820
TCustomRVFontInfo 820
TFontInfos 788
TParaInfos 793
TRVListInfos 797
AssignToRVPrint 1040
TRVReportHelperWithHeaderFooters 1040
AssignToStrings 799
TRVStyleTemplateCollection 799
Attached files 195
Attachment 1022
TRVBackgroundProperties 1022
Author 424
TRVDocParameters 424
AutoDeleteUnusedStyles 430, 608, 621
TDBRichView 608
TDBRichViewEdit 621
TRVDocumentProperty 430
AutoDestroy 670
TRVChatMessageExtra 670

AutoDisplay 609, 622
TDBRichView 609
TDBRichViewEdit 622

AutoHideTableGridLines 459
TRVRTFReaderProperties 459

AvailableLanguages 731
TRVSpellChecker 731

- B -

B 680
TRVChatUITexts 680

BackColor 660, 808
TCustomRVFontInfo 808
TRVChatDateProperties 660

Background 830, 995
TCustomRVParaInfo 830
TRVBoxProperties 995

BackgroundBitmap 229
TRichView 229

BackgroundImage 904, 953
TRVTableCellData 953
TRVTableItemInfo 904

BackgroundImageFileName 905, 953
TRVTableCellData 953
TRVTableItemInfo 905

BackgroundPicture 229
TRichView 229

BackgroundProperties 229, 905, 954
TRichView 229
TRVTableCellData 954
TRVTableItemInfo 905

BackgroundPropertiesAllocate 964
TRVTableCellData 964

BackgroundPropertiesFinalize 964
TRVTableCellData 964

BackgroundStyle 229, 905, 954
TRichView 229
TRVTableCellData 954
TRVTableItemInfo 905

BasePathLinks 433, 453, 459
TRVHTMLReaderProperties 433
TRVRTFReaderProperties 459

BeginItemModify 503
TRichViewEdit 503

BeginOleDrag 284
TRichView 284

BeginUndoCustomGroup 504
TRichViewEdit 504

BeginUndoCustomGroup2 504
TRichViewEdit 504

BeginUndoGroup 505
TRichViewEdit 505

BeginUndoGroup2 505
TRichViewEdit 505

BeginUpdate 506
TRichViewEdit 506

BestDPI 702
TRVPrint 702

BestHeight 955
TRVTableCellData 955

BestWidth 906, 955
TRVTableCellData 955
TRVTableItemInfo 906

BiDiMode 137, 230, 808, 831
TCustomRVFontInfo 808
TCustomRVParaInfo 831
TRichView 230

Bidirectional text in TRichView 137

BlockQuoteBackColor 446
TRVMarkdownDefaultItemProperties 446

Border 831, 995
TCustomRVParaInfo 831
TRVBoxProperties 995

BorderColor 906, 956
TRVTableCellData 956
TRVTableItemInfo 906

BorderHSpacing 907
TRVTableItemInfo 907

BorderLightColor 907, 956
TRVTableCellData 956
TRVTableItemInfo 907

BorderOffsets 851, 853
TRVBackgroundRect 851
TRVBorder 853

Borders and background of paragraphs in TRichView 102

BorderStyle 871, 908
TRVScroller 871
TRVTableItemInfo 908

BorderVSpacing 908
TRVTableItemInfo 908

BorderWidth 909
TRVTableItemInfo 909

BottomMargin 231, 424
TRichView 231
TRVDocParameters 424

BoxPosition 993
TRVSidenoteItemInfo 993

BoxProperties 992

BoxProperties 992
TRVSidenoteItemInfo 992
BreakColor 446
TRVMarkdownDefaultItemProperties 446
Breaks 173
Building TRichView document 107
Bullets 176
ButtonType 599
TRVSmartPopupProperties 599

- C -

CachePageImage 638
TRVPrintPreview 638
Callout 641
TRVChat 641
CalloutType 658
TRVChatCalloutProperties 658
CanChange 506
TRichViewEdit 506
CanMergeCells 925
TRVTableItemInfo 925
CanMergeSelectedCells 926
TRVTableItemInfo 926
CanPaste 507
TRichViewEdit 507
CanPasteHTML 507
TRichViewEdit 507
CanPasteRTF 507
TRichViewEdit 507
CanPasteRVF 507
TRichViewEdit 507
CanReuseNumberedLists 459
TRVRTFReaderProperties 459
CanSavePDF 883
TCustomRVPrint 883
Cell Merging 204
CellBorderColor 909
TRVTableItemInfo 909
CellBorderLightColor 910
TRVTableItemInfo 910
CellBorderStyle 910
TRVTableItemInfo 910
CellBorderWidth 910
TRVTableItemInfo 910
CellHPadding 911
TRVTableItemInfo 911
CellHSpacing 911
TRVTableItemInfo 911
CellOverrideColor 912
TRVTableItemInfo 912
CellPadding 912
TRVTableItemInfo 912
Cells 912
TRVTableItemInfo 912
CellVPadding 913
TRVTableItemInfo 913
CellVSpacing 913
TRVTableItemInfo 913
CFRV_HTML 1104
CFRV_RTF 1104
CFRV_RVF 1104
CFRV_URL 1104
Change 508, 631
TDBRichViewEdit 631
TRichViewEdit 508
Changed 927
TRVTableItemInfo 927
ChangeListLevels 508
TRichViewEdit 508
ChangeMessageImage 646
TRVChat 646
ChangeMessageText 647
TRVChat 647
ChangeStyleTemplates 508
TRichViewEdit 508
CharScale 808
TCustomRVFontInfo 808
Charset 809
TCustomRVFontInfo 809
CharsetForUnicode 460
TRVRTFReaderProperties 460
CharSpacing 809
TCustomRVFontInfo 809
ChatMaxWidth 641
TRVChat 641
ChatRichView 642
TRVChat 642
CheckpointColor 743
TRVStyle 743
CheckpointEvColor 743
TRVStyle 743
Checkpoints in TRichView 92
CheckpointsPrefix 437
TRVHTMLSaveProperties 437
CheckSpelling 480
TRichViewEdit 480
Clear 286, 509, 883
TCustomRVPrint 883
TRichView 286

Clear 286, 509, 883
TRichViewEdit 509
ClearLeft 232
TRichView 232
ClearLiveSpellingResults 286
TRichView 286
ClearMessages 647
TRVChat 647
ClearParaFormat 800
TRVStyleTemplateCollection 800
ClearRight 232
TRichView 232
ClearSoftPageBreaks 286
TRichView 286
ClearTextFlow 509
TRichViewEdit 509
ClearTextFormat 800
TRVStyleTemplateCollection 800
ClearUndo 509
TRichViewEdit 509
ClickMode 891
TCustomRVPrintPreview 891
ClientToDocument 286
TRichView 286
ClipMargins 703
TRVPrint 703
CodeBlockBackColor 446
TRVMarkdownDefaultItemProperties 446
CodePage 431, 609, 622
TDBRichView 609
TDBRichViewEdit 622
TRVDocumentProperty 431
ColBandSize 913
TRVTableItemInfo 913
ColCount 914
TRVTableItemInfo 914
Color 233, 599, 744, 810, 852, 853, 891, 914, 957
TCustomRVFontInfo 810
TCustomRVPrintPreview 891
TRichView 233
TRVBackgroundRect 852
TRVBorder 853
TRVSmartPopupProperties 599
TRVStyle 744
TRVTableCellData 957
TRVTableItemInfo 914
ColorMode 878
TCustomRVPrint 878
Colors and layout of tables 199
ColSpan 957
TRVTableCellData 957
Comments 424
TRVDocParameters 424
CompactQuotes 679
TRVChatQuoteProperties 679
Component Editor for TRichView 224
ContinuousPrint 709
TRVPrint 709
Controls 174
Controls - Documents - Items 141
ConvertDocToDifferentUnits 287
TRichView 287
ConvertDocToEMU 287
TRichView 287
ConvertDocToPixels 287
TRichView 287
ConvertDocToTwips 287
TRichView 287
ConvertHighlight 460
TRVRTFReaderProperties 460
ConvertSymbolFonts 461
TRVRTFReaderProperties 461
ConvertToDifferentUnits 767
TRVStyle 767
ConvertToEMU 767
TRVStyle 767
ConvertToHotPicture 510
TRichViewEdit 510
ConvertToPicture 510
TRichViewEdit 510
ConvertToPixels 767
TRVStyle 767
ConvertToTwips 767
TRVStyle 767
ConvertToUnit 428
TRVDocParameters 428
ConvertToUnits 710
TRVPrint 710
Copy 288
TRichView 288
CopyDef 288
TRichView 288
CopyImage 289
TRichView 289
CopyMarkdown 289
TRichView 289
CopyRTF 290
TRichView 290
CopyRVF 290
TRichView 290

CopyText 290
TRichView 290
CopyTextA 290
TRichView 290
CopyTextW 290
TRichView 290
CopyToClipboard 647
TRVChat 647
CornerRadius 680
TRVChatQuoteProperties 680
CornerRadiusLarge 658
TRVChatCalloutProperties 658
CornerRadiusSmall 658
TRVChatCalloutProperties 658
Count 696
TRVOfficeCnvList 696
CPEventKind 233
TRichView 233
Create 291, 510, 614, 631, 672, 673, 674, 692, 710, 767, 788, 794, 821, 825, 836, 840, 875, 897, 927, 973, 978, 982, 987, 990, 993, 1005, 1008, 1010, 1012, 1040
TCustomRVFontInfo 821
TCustomRVParaInfo 836
TCustomRVPrintPreview 897
TDBRichView 614
TDBRichViewEdit 631
TFontInfo 825
TFontInfos 788
TParaInfo 840
TParaInfos 794
TRichView 291
TRichViewEdit 510
TRVChatMessageExtraFiles 672
TRVChatMessageExtraImage 673
TRVChatMessageExtraQuote 674
TRVEndnoteItemInfo 987
TRVFootnoteItemInfo 990
TRVLabelItemInfo 973
TRVMathItemInfo 978
TRVNoteReferenceItemInfo 1008
TRVOfficeConverter 692
TRVPageCountItemInfo 1012
TRVPageNumberItemInfo 1010
TRVPrint 710
TRVReportHelperWithHeaderFooters 1040
TRVScroller 875
TRVSeqItemInfo 982
TRVSidenoteItemInfo 993
TRVStyle 767
TRVTableItemInfo 927
TRVTextBoxItemInfo 1005
CreateEx 928, 973, 982, 984, 987, 990, 993, 1005, 1008, 1011, 1012
TCustomRVNoteItemInfo 984
TRVEndnoteItemInfo 987
TRVFootnoteItemInfo 990
TRVLabelItemInfo 973
TRVNoteReferenceItemInfo 1008
TRVPageCountItemInfo 1012
TRVPageNumberItemInfo 1011
TRVSeqItemInfo 982
TRVSidenoteItemInfo 993
TRVTableItemInfo 928
TRVTextBoxItemInfo 1005
crJump 147
crRVZoomIn 147
crRVZoomOut 147
CSSOptions 438
TRVHTMLSaveProperties 438
CSSSavingType 438
TRVHTMLSaveProperties 438
CurItemNo 480
TRichViewEdit 480
CurItemStyle 481
TRichViewEdit 481
CurParaStyleNo 481
TRichViewEdit 481
CurrentItemColor 744
TRVStyle 744
CurrentLanguage 733
TRVSpellChecker 733
Cursor 234, 971
TRichView 234
TRVLabelItemInfo 971
Cursors in TRichView 147
CurTextStyleNo 482
TRichViewEdit 482
Custom drawing in TRichView 146
Custom item types in TRichView 195
CustomCaretInterval 483
TRichViewEdit 483
CutDef 511
TRichViewEdit 511

- D -

DarkMode 234, 878
TCustomRVPrint 878
TRichView 234
DarkModeUI 892
TCustomRVPrintPreview 892

DataField 610, 622
TDBRichView 610
TDBRichViewEdit 622
DataSource 610, 623, 686
TDBRichView 610
TDBRichViewEdit 623
TRVDataSourceLink 686
DateFont 664
TRVChatFonts 664
DateFormatNew 660
TRVChatDateProperties 660
DateFormatOld 660
TRVChatDateProperties 660
DateLine 642
TRVChat 642
DateTime 666
TRVChatMessage 666
Default0Style 439
TRVHTMLSaveProperties 439
DefaultBulletFontName 1026
TRVDefaultLoadProperties 1026
DefaultFontName 1026
TRVDefaultLoadProperties 1026
DefaultFontSizeDouble 1027
TRVDefaultLoadProperties 1027
DefaultFontSizesDouble 1027
TRVDefaultLoadProperties 1027
DefaultItemProperties 452
TRVMarkdownProperties 452
DefaultMonoSpacedFontName 1028
TRVDefaultLoadProperties 1028
DefaultPictureVAlign 483
TRichViewEdit 483
DefaultProperties 1028
TRVDefaultLoadProperties 1028
DefaultRVFPixelsPerInch 1109
DefaultUnicodeSymbolFontName 461, 1028
TRVDefaultLoadProperties 1028
DefCharsetCodePage 461
TRVRTFReaderProperties 461
DefCodePage 745
TRVStyle 745
DefStyleName 798
TRVStyleTemplateCollection 798
DefStyleNo 839
TParaInfo 839
DefTabWidth 745
TRVStyle 745
DefUnicodeStyle 746
TRVStyle 746
DeleteCols 928
TRVTableItemInfo 928
DeletedMessage 680
TRVChatUITexts 680
DeletedMessageFont 664
TRVChatFonts 664
DeleteEmptyCols 929
TRVTableItemInfo 929
DeleteEmptyRows 929
TRVTableItemInfo 929
DeleteItems 291
TRichView 291
DeleteList 855
TRVTabInfos 855
DeleteMarkedStyles 292
TRichView 292
DeleteMessage 647
TRVChat 647
DeleteParas 293
TRichView 293
DeleteRows 929
TRVTableItemInfo 929
DeleteSection 293
TRichView 293
DeleteSelectedCols 930
TRVTableItemInfo 930
DeleteSelectedRows 930
TRVTableItemInfo 930
DeleteSelection 511
TRichViewEdit 511
DeleteTextStyle 767
TRVStyle 767
DeleteUnusedStyles 293
TRichView 293
Delimiters 235
TRichView 235
Deselect 294, 931
TRichView 294
TRVTableItemInfo 931
Destroy 294, 511, 614, 631, 692, 767, 931
TDBRichView 614
TDBRichViewEdit 631
TRichView 294
TRichViewEdit 511
TRVOfficeConverter 692
TRVStyle 767
TRVTableItemInfo 931
DialogShown 731
TRVSpellChecker 731
DifferentUnitsToUnits 769

DifferentUnitsToUnits 769
TRVStyle 769
DisabledFontColor 746
TRVStyle 746
DisplayInline 975
TRVMathItemInfo 975
DisplayName 664
TRVChatFileItem 664
DocObjects 235
TRichView 235
DocParameters 237
TRichView 237
DocProperties 238
TRichView 238
Document 238, 984
TCustomRVNoteItemInfo 984
TRichView 238
DocumentHeight 239
TRichView 239
DocumentPixelsPerInch 239
TRichView 239
DocumentToClient 286
TRichView 286
DocXErrorCode 462
TRVRTFReaderProperties 462
DoInPaletteMode 241
TRichView 241
Drag and drop in TRichView 123
Drag&Drop in TRichView 123
Draw 825
TFontInfo 825
DrawButton 1043
DrawControl 1043
DrawEdit 1043
DrawMemo 1043
DrawPage 710, 725
TRVPrint 710
TRVReportHelper 725
DrawPageAt 710, 725
TRVPrint 710
TRVReportHelper 725
DrawPanel 1043
DrawPreview 711
TRVPrint 711

- E -

Edit 1016
TCustomRVData 1016
EditCell 931
TRVTableItemInfo 931
EditCount 666
TRVChatMessage 666
Edited 680
TRVChatUITexts 680
Editing Cells 203
Editor 686
TRVDataSourceLink 686
EditorOptions 483
TRichViewEdit 483
EmptyWidth 810
TCustomRVFontInfo 810
EMUToUnits 769
TRVStyle 769
Encoding 439
TRVHTMLSaveProperties 439
EncodingStr 439
TRVHTMLSaveProperties 439
EndAt 878
TCustomRVPrint 878
EndItemModify 511
TRichViewEdit 511
EndnoteNumbering 746
TRVStyle 746
Endnotes 184
EndUndoCustomGroup2 504
TRichViewEdit 504
EndUndoGroup2 505
TRichViewEdit 505
EndUpdate 512
TRichViewEdit 512
EnforceListLevels 433
TRVHTMLReaderProperties 433
equation objects 1043
ERichViewError 1014
ErrorCode 689
TRVOfficeConverter 689
EvenColumnsColor 923
TRVTableItemInfo 923
EvenRowsColor 923
TRVTableItemInfo 923
Example
Converting table to text 212
Dividing table 214
Inserting controls what can modify themselves in DBRichViewEdit 634
Loading UTF-8 files 468
Moving caret to the beginning of paragraph 602
OnDropFiles example 603
Using viewer-style methods in DBRichViewEdit 634

ExcludeDocXExportConverter 690
TRVOfficeConverter 690
ExcludeDocXImportConverter 690
TRVOfficeConverter 690
ExcludeHTMLExportConverter 690
TRVOfficeConverter 690
ExcludeHTMLImportConverter 690
TRVOfficeConverter 690
Execute 687, 733
TRVDataSourceLink 687
TRVSpellChecker 733
Export of Tables 209
Export of TRichView contents to HTML 132
ExportConverters 690
TRVOfficeConverter 690
ExportRTF 692
TRVOfficeConverter 692
ExportRV 693
TRVOfficeConverter 693
Extensions 452
TRVMarkdownProperties 452
ExtensionsInFilter 690
TRVOfficeConverter 690
ExternalCSSFileName 440
TRVHTMLSaveProperties 440
ExtractMetafileBitmaps 462
TRVRTFReaderProperties 462
ExtraStyles 440
TRVHTMLSaveProperties 440

- F -

FacingPages 424, 703
TRVDocParameters 424
TRVPrint 703
FailedImageIndex 679
TRVChatMessageStatusLine 679
Field 610, 623
TDBRichView 610
TDBRichViewEdit 623
FieldFormat 431, 623
TDBRichViewEdit 623
TRVDocumentProperty 431
FieldHighlightColor 746
TRVStyle 746
FieldHighlightType 747
TRVStyle 747
FileIcons 642
TRVChat 642
FileInfoFont 664
TRVChatFonts 664
FileName 664
TRVChatFileItem 664
FileNameFont 664
TRVChatFonts 664
Files 668, 671
TRVChatMessage 668
TRVChatMessageExtraFiles 671
FileSize 664
TRVChatFileItem 664
FileType 664
TRVChatFileItem 664
Fill 891
TCustomRVPrintPreview 891
FillColor 682
TRVChatUser 682
FillLanguageNames 734
TRVSpellChecker 734
Filter 696
TRVOfficeConverterInfo 696
Find 855
TRVTabInfos 855
FindByDate 669
TRVChatMessageCollection 669
FindByGlobalIndex 669
TRVChatMessageCollection 669
FindById 669, 684, 800
TRVChatMessageCollection 669
TRVChatUserCollection 684
TRVStyleTemplateCollection 800
FindByName 800
TRVStyleTemplateCollection 800
FindCheckpointByName 294
TRichView 294
FindCheckpointByTag 295
TRichView 295
FindControlItemNo 295
TRichView 295
FindInsertionPlace 670
TRVChatMessageCollection 670
FindItemByDate 669
TRVChatMessageCollection 669
FindItemByGlobalIndex 669
TRVChatMessageCollection 669
FindItemById 669, 684, 800
TRVChatMessageCollection 669
TRVChatUserCollection 684
TRVStyleTemplateCollection 800
FindItemByName 800
TRVStyleTemplateCollection 800

FindMessages 648
TRVChat 648
FindParaStyle 771
TRVStyle 771
FindStyleWithAlignment 794
TParaInfos 794
FindStyleWithCharset 788
TFontInfos 788
FindStyleWithColor 788
TFontInfos 788
FindStyleWithFont 789
TFontInfos 789
FindStyleWithFontName 789
TFontInfos 789
FindStyleWithFontSize 790
TFontInfos 790
FindStyleWithFontStyle 790
TFontInfos 790
FindStyleWithLevels 796
TRVListInfos 796
FindSuchStyle 791, 794, 796
TFontInfos 791
TParaInfos 794
TRVListInfos 796
FindSuchStyleEx 791, 794
TFontInfos 791
TParaInfos 794
FindTextStyle 771
TRVStyle 771
Finished 726
TRVReportHelper 726
FireMonkey 89, 161
First 897
TCustomRVPrintPreview 897
FirstColumnColor 923
TRVTableItemInfo 923
FirstIndent 831, 860
TCustomRVParaInfo 831
TRVListLevel 860
FirstItemVisible 243
TRichView 243
FirstJumpNo 243
TRichView 243
FirstPageNo 879
TCustomRVPrint 879
FixMarginsMode 703
TRVPrint 703
FloatingLineColor 747
TRVStyle 747
Font 861
TRVListLevel 861
FontName 811, 976
TCustomRVFontInfo 811
TRVMathItemInfo 976
FontQuality 748
TRVStyle 748
Fonts 642
TRVChat 642
FontSizeDouble 976
TRVMathItemInfo 976
FontSizePrecision 434
TRVHTMLReaderProperties 434
FooterY 425, 704
TRVDocParameters 425
TRVPrint 704
FootnoteNumbering 746
TRVStyle 746
FootnotePageReset 749
TRVStyle 749
Footnotes 185
ForbiddenIds 798
TRVStyleTemplateCollection 798
ForceFieldHighlight 486
TRichViewEdit 486
Format 295
TRichView 295
FormatNextPage 726
TRVReportHelper 726
FormatPages 711
TRVPrint 711
FormatString 861, 980
TRVListLevel 861
TRVSeqItemInfo 980
FormatStringW 862
TRVListLevel 862
FormatTail 296
TRichView 296
FullRedraw 750
TRVStyle 750

- G -

GB 680
TRVChatUITexts 680
GenericFontNames 1028
TRVDefaultLoadProperties 1028
GetAllText 1045
GetAsDifferentUnits 768
TRVStyle 768
GetAsEMU 768

GetAsEMU 768
TRVStyle 768
GetAsHTMLPixels 768
TRVStyle 768
GetAsPixels 768
TRVStyle 768
GetAsRVUnits 768
TRVStyle 768
GetAsStandardPixels 768
TRVStyle 768
GetAsTwips 768
TRVStyle 768
GetAutoCorrection 734
TRVSpellChecker 734
GetAutoQuote 648
TRVChat 648
GetAutoQuoteFromItemNo 648
TRVChat 648
GetBreakInfo 296
TRichView 296
GetBulletInfo 297
TRichView 297
GetCellAt 932
TRVTableItemInfo 932
GetCheckpointAtCaret 512
TRichViewEdit 512
GetCheckpointByNo 298
TRichView 298
GetCheckpointInfo 299
TRichView 299
GetCheckpointItemNo 299
TRichView 299
GetCheckpointNo 299
TRichView 299
GetCheckpointXY 300
TRichView 300
GetCheckpointY 300
TRichView 300
GetCheckpointYEx 301
TRichView 301
GetControlInfo 301
TRichView 301
GetCurrentBreakInfo 512
TRichViewEdit 512
GetCurrentBulletInfo 513
TRichViewEdit 513
GetCurrentChar 1045
GetCurrentCheckpoint 514
TRichViewEdit 514
GetCurrentControlInfo 515
TRichViewEdit 515
GetCurrentHotspotInfo 515
TRichViewEdit 515
GetCurrentItem 517
TRichViewEdit 517
GetCurrentItemEx 517
TRichViewEdit 517
GetCurrentItemExtraIntProperty 518
GetCurrentItemExtraIntPropertyEx 518
TRichViewEdit 518
GetCurrentItemExtraStrProperty 519
TRichViewEdit 519
GetCurrentItemExtraStrPropertyEx 519
TRichViewEdit 519
GetCurrentItemText 519
TRichViewEdit 519
GetCurrentItemTextA 519
TRichViewEdit 519
GetCurrentItemTextW 519
TRichViewEdit 519
GetCurrentItemVAlign 520
TRichViewEdit 520
GetCurrentLineCol 521
TRichViewEdit 521
GetCurrentLineText 1045
GetCurrentMisspelling 521
TRichViewEdit 521
GetCurrentParaSectionText 1045
GetCurrentParaText 1045
GetCurrentPictureInfo 522
TRichViewEdit 522
GetCurrentTag 522
TRichViewEdit 522
GetCurrentTextInfo 523
TRichViewEdit 523
GetCurrentWord 1045
GetEditedCell 932
TRVTableItemInfo 932
GetExportFilter 693
TRVOfficeConverter 693
GetFirstCheckpoint 302
TRichView 302
GetFirstItemOnPage 883
TCustomRVPrint 883
GetFirstVisibleMessage 649
TRVChat 649
GetFocusedItem 302
TRichView 302
GetFooter 1040
TRVReportHelperWithHeaderFooters 1040

GetFooterRect 712
TRVPrint 712

GetHeader 1040
TRVReportHelperWithHeaderFooters 1040

GetHeaderRect 712
TRVPrint 712

GetHotspotInfo 303
TRichView 303

GetImageAtCursor 649
TRVChat 649

GetImportFilter 693
TRVOfficeConverter 693

GetItem 304
TRichView 304

GetItemAt 305
TCustomRichView 305

GetItemCheckpoint 305
TRichView 305

GetItemClientCoords 306
TRichView 306

GetItemCoords 306
TRichView 306

GetItemCoordsEx 307
TRichView 307

GetItemExtraIntProperty 307
TRichView 307

GetItemExtraIntPropertyEx 307
TRichView 307

GetItemExtraStrProperty 308
TCustomRichView 308

GetItemExtraStrPropertyEx 308
TCustomRichView 308

GetItemNo 309
TRichView 309

GetItemPara 309
TRichView 309

GetItemStyle 310
TRichView 310

GetItemTag 310
TRichView 310

GetItemText 310
TRichView 310

GetItemTextA 310
TRichView 310

GetItemTextW 310
TRichView 310

GetItemVAlign 311
TRichView 311

GetJumpPointItemNo 312
TRichView 312

GetJumpPointLocation 312
TRichView 312

GetJumpPointY 313
TRichView 313

GetLanguageNameFromCode 735
TRVSpellChecker 735

GetLastCheckpoint 313
TRichView 313

GetLastPageHeight 726
TRVReportHelper 726

GetLineNo 314
TRichView 314

GetListMarkerInfo 315
TRichView 315

GetListStyleClass 770
TRVStyle 770

GetMainCell 969
TRVTableRows 969

GetMessage 649
TRVChat 649

GetMessageAndImage 649
TRVChat 649

GetMessageAtClientCoords 649
TRVChat 649

GetMessageAtCursor 649
TRVChat 649

GetMessageAtDocumentCoords 649
TRVChat 649

GetMessageAtScreenCoords 649
TRVChat 649

GetMessageImage 650
TRVChat 650

GetMessageImageEx 650
TRVChat 650

GetMessageItemNo 650
TRVChat 650

GetMessagesAsText 651
TRVChat 651

GetMessageTextItemNo 650
TRVChat 650

GetNextCheckpoint 315
TRichView 315

GetNextFileName 383

GetNormalizedSelectionBounds 933
TRVTableItemInfo 933

GetOffsAfterItem 316
TRichView 316

GetOffsBeforeItem 317
TRichView 317

GetOriginEx 1018

GetOriginEx 1018
TCustomRVFormattedData 1018
GetPageNo 883
TCustomRVPrint 883
GetParaStyleClass 770
TRVStyle 770
GetPictureInfo 318
TRichView 318
GetPrevCheckpoint 319
TRichView 319
GetRangeOfSelectedMessages 651
TRVChat 651
GetRealDocumentPixelsPerInch 319
TRichView 319
GetRVData 964, 1017
TCustomRVData 1017
TRVTableCellData 964
GetRVDataText 1045
GetRVESearchOptions 1049
GetRVSearchOptions 1049
GetSelectedImage 319
TRichView 319
GetSelectedMessage 651
TRVChat 651
GetSelectedTextForQuote 652
TRVChat 652
GetSelectionBounds 320, 933
TRichView 320
TRVTableItemInfo 933
GetSelText 321
TRichView 321
GetSelTextA 321
TRichView 321
GetSelTextW 321
TRichView 321
GetSourceRVData 1017
TCustomRVData 1017
GetTextInfo 321
TRichView 321
GetTextStyleClass 770
TRVStyle 770
GetVisibleText 1045
GetWordAt 322
TRichView 322
GetWordAtA 322
TRichView 322
GetWordAtW 322
TRichView 322
GlobalIndex 667
TRVChatMessage 667
GridColor 750
TRVStyle 750
GridReadOnlyColor 750
TRVStyle 750
GridReadOnlyStyle 750
TRVStyle 750
GridStyle 750
TRVStyle 750
GuessCompletions 735
TRVSpellChecker 735

- H -

HasNumbering 843, 868
TRVListInfo 843
TRVListLevel 868
HasSelectedTextForQuote 652
TRVChat 652
HeaderY 425, 705
TRVDocParameters 425
TRVPrint 705
HeadingRowColor 923
TRVTableItemInfo 923
HeadingRowCount 914
TRVTableItemInfo 914
Height 995
TRVBoxProperties 995
HeightType 995
TRVBoxProperties 995
Hidden text in TRichView 106
HighlightedMessage 642
TRVChat 642
Hint 600, 958
TRVSmartPopupProperties 600
TRVTableCellData 958
HorizontalAlignment 999, 1023
TRVBackgroundProperties 1023
TRVBoxPosition 999
HorizontalAnchor 998
TRVBoxPosition 998
HorizontalOffset 999, 1023
TRVBackgroundProperties 1023
TRVBoxPosition 999
HorizontalOffsetKind 1023
TRVBackgroundProperties 1023
HorizontalPositionKind 999
TRVBoxPosition 999
HorizontalRepeat 1024
TRVBackgroundProperties 1024
HorizontalSizeKind 1024

HorizontalSizeKind 1024
TRVBackgroundProperties 1024
Hot-Pictures 172
Hotspots 178
HOutermostRule 915
TRVTableItemInfo 915
HoverBackColor 811
TCustomRVFontInfo 811
HoverColor 600, 750, 812
TCustomRVFontInfo 812
TRVSmartPopupProperties 600
TRVStyle 750
HoverEffects 812
TCustomRVFontInfo 812
HoverLineColor 600
TRVSmartPopupProperties 600
HoverUnderlineColor 813
TCustomRVFontInfo 813
How to
(List) 1130
change page size and orientation for printing RichView 1131
combine several TRichView documents in one 1136
copy one TRichView to another 282
create a chat window 1139
implement commands like "make bold", "apply font" 1131
implement Find and Replace dialogs 1132
implement headers and footers 1131
implement smart indents 1136
make a plain text editor 1134
make Unicode editor 1134
move a caret to the beginning or to the end of document in TRichViewEdit 1133
remove text formatting 1139
switch insert/overtype mode 1132
use third-party graphic classes with RichView 1136
HRuleColor 915
TRVTableItemInfo 915
HRuleWidth 916
TRVTableItemInfo 916
HScrollMax 871
TRVScroller 871
HScrollPos 872
TRVScroller 872
HScrollVisible 872
TRVScroller 872
HTML and TRichView 132
HTMLFormatName 1104
HTMLPixelsToUnits 769
TRVStyle 769
HTMLReadProperties 243
TRichView 243
HTMLSaveProperties 244
TRichView 244
HTMLSavingType 441
TRVHTMLSaveProperties 441
HTMLVersion 441
TRVHTMLSaveProperties 441
Hypertext in TRichView 97

- I -

IconColor 662
TRVChatDateProperties 662
IconPadding 662
TRVChatDateProperties 662
IconSize 662
TRVChatDateProperties 662
Id 666, 682, 844
TRVChatMessage 666
TRVChatUser 682
TRVStyleTemplate 844
IDAsCheckpoints 434
TRVHTMLReaderProperties 434
IgnoreBookmarks 463
TRVRTFReaderProperties 463
IgnoreContentHeight 958
TRVTableCellData 958
IgnoreEscape 432, 624
TDBRichViewEdit 624
TRVDocumentProperty 432
IgnoreFields 463
TRVRTFReaderProperties 463
IgnoreNotes 463
TRVRTFReaderProperties 463
IgnorePageBreaks 879
TCustomRVPrint 879
IgnorePictures 463
TRVRTFReaderProperties 463
IgnoreSequences 463
TRVRTFReaderProperties 463
IgnoreTables 464
TRVRTFReaderProperties 464
Image 673
TRVChatMessageExtraImage 673
ImageHeight 863, 1025
TRVBackgroundProperties 1025
TRVListLevel 863

ImageIndex 600, 863
TRVListLevel 863
TRVSmartPopupProperties 600
ImageList 600, 863
TRVListLevel 863
TRVSmartPopupProperties 600
ImageOptions 442
TRVHTMLSaveProperties 442
Images 677
TRVChatMessageStatusLine 677
ImagesHeight 677
TRVChatMessageStatusLine 677
ImagesPrefix 443, 453
TRVHTMLSaveProperties 443
TRVMarkdownProperties 453
ImagesWidth 677
TRVChatMessageStatusLine 677
ImageText 673
TRVChatMessageExtraImage 673
ImageVAlign 447
TRVMarkdownDefaultItemProperties 447
ImageWidth 863, 1024
TRVBackgroundProperties 1024
TRVListLevel 863
imgSaveNo 383
ImportConverters 691
TRVOfficeConverter 691
ImportRTF 694
TRVOfficeConverter 694
ImportRV 694
TRVOfficeConverter 694
InactiveSelColor 751
TRVStyle 751
InactiveSelTextColor 751
TRVStyle 751
Init 727
TRVReportHelper 727
InplaceEditor 872
TRVScroller 872
InsertBreak 523
TRichViewEdit 523
InsertBullet 524
TRichViewEdit 524
InsertCheckpoint 525
TRichViewEdit 525
InsertCols 934
TRVTableItemInfo 934
InsertColsLeft 934
TRVTableItemInfo 934
InsertColsRight 935
TRVTableItemInfo 935
InsertControl 526
TRichViewEdit 526
InsertDocXFromFileEd 526
TRichViewEdit 526
InsertDocXFromStreamEd 527
TRichViewEdit 527
InsertFromRVST 801
TRVStyleTemplateCollection 801
InsertHotPicture 528
TRichViewEdit 528
InsertHotspot 528
TRichViewEdit 528
InsertHTMLFromFileEd 529
TRichViewEdit 529
InsertHTMLFromStreamEd 530
TRichViewEdit 530
InsertItem 531
TRichViewEdit 531
InsertMarkdownFromFileEd 532
TRichViewEdit 532
InsertMarkdownFromStreamEd 532
TRichViewEdit 532
InsertMessage 652
TRVChat 652
InsertOEMTextFromFile 533
TRichViewEdit 533
InsertPageBreak 534
TRichViewEdit 534
InsertPicture 534
TRichViewEdit 534
InsertRows 935
TRVTableItemInfo 935
InsertRowsAbove 936
TRVTableItemInfo 936
InsertRowsBelow 936
TRVTableItemInfo 936
InsertRTFFromFileEd 536
TRichViewEdit 536
InsertRTFFromStreamEd 536
TRichViewEdit 536
InsertRVFFromFileEd 537
TRichViewEdit 537
InsertRVFFromStream 323
TRichView 323
InsertRVFFromStreamEd 538
TRichViewEdit 538
InsertStringATag 539
TRichViewEdit 539
InsertStringTag 539

InsertStringTag 539
TRichViewEdit 539
InsertStringWTag 539
TRichViewEdit 539
InsertTab 540
TRichViewEdit 540
InsertText 540
TRichViewEdit 540
InsertTextA 540
TRichViewEdit 540
InsertTextFromFile 541
TRichViewEdit 541
InsertTextFromFileW 541
TRichViewEdit 541
InsertTextW 540
TRichViewEdit 540
InternalWidth 853
TRVBorder 853
Intersect 856
TRVTabInfos 856
InvalidItem 786, 792
TFontInfos 786
TParaInfos 792
InvalidPicture 752
TRVStyle 752
IsAvailable 735
TRVSpellChecker 735
IsCellSelected 936
TRVTableItemInfo 936
IsEqual 821, 827, 837, 841
TCustomRVFontInfo 821
TCustomRVParaInfo 837
TFontInfo 827
TParaInfo 841
IsFromNewLine 325
TRichView 325
IsParaStart 325
TRichView 325
IsRightSide 682
TRVChatUser 682
IsValidImporter 695
TRVOfficeConverter 695
IsWordValid 735
TRVSpellChecker 735
Item types
Attached files 195
Breaks 173
Bullets 176
Controls 174
Endnotes 184
Footnotes 185
Hot-Pictures 172
Hotspots 178
Labels 181
List Markers 180
Mathematical expressions 193
Numbered sequences 183
Page counts 192
Page numbers 190
Pictures 169
References to parent endnotes and footnotes 190
Sidenotes 186
Tables 179
Tabulators 168
Text 167
Text boxes 188
Item types in TRichView 163, 899
ItemCount 244
TRichView 244
Items 662, 668, 684, 696, 786, 792, 795, 798, 855, 859, 967, 969
TFontInfos 786
TParaInfos 792
TRVChatFileCollection 662
TRVChatMessageCollection 668
TRVChatUserCollection 684
TRVListInfos 795
TRVListLevelCollection 859
TRVOfficeCnvList 696
TRVStyleTemplateCollection 798
TRVTabInfos 855
TRVTableRow 967
TRVTableRows 969

- J -

Jump 823
TFontInfo 823
JumpCursor 752, 813
TCustomRVFontInfo 813
TRVStyle 752

- K -

KB 680
TRVChatUITexts 680
KeepTogether 967
TRVTableRow 967
KeyboardType 487
TRichViewEdit 487
Keys processed by tables 212

Kind 845
TRVStyleTemplate 845

- L -

Labels 181
Language 731
TRVSpellChecker 731
LanguageIndex 732
TRVSpellChecker 732
Last 897
TCustomRVPrintPreview 897
LastColumnColor 923
TRVTableItemInfo 923
LastItemVisible 244
TRichView 244
LastLineAlignment 829
TCustomRVParaInfo 829
LastPageNo 879
TCustomRVPrint 879
LastRowColor 923
TRVTableItemInfo 923
Lazarus 160
Leader 857
TRVTabInfo 857
LeftIndent 832, 864
TCustomRVParaInfo 832
TRVListLevel 864
LeftMargin 245, 425
TRichView 245
TRVDocParameters 425
Levels 842
TRVListInfo 842
LinearToRichView 1047
LineBreaksAsParagraphs 464
TRVRTFReaderProperties 464
LineColor 601
TRVSmartPopupProperties 601
LineCount 245
TRichView 245
LineSelectCursor 753
TRVStyle 753
LineSpacing 832
TCustomRVParaInfo 832
LineSpacingType 833
TCustomRVParaInfo 833
LineWidth 453
TRVMarkdownProperties 453
LineWrapMode 753
TRVStyle 753
LinkColor 643
TRVChat 643
List Markers 180
ListLeftIndentPix 447
TRVMarkdownDefaultItemProperties 447
ListMarkerIndentPix 447, 1031
TRVDefaultLoadProperties 1031
TRVMarkdownDefaultItemProperties 447
ListMarkerSavingType 444
TRVHTMLSaveProperties 444
ListStyles 755
TRVStyle 755
ListType 864
TRVListLevel 864
Live spelling check in TRichView 138
LiveSpellingColor 755
TRVStyle 755
LiveSpellingMode 487
TRichViewEdit 487
LiveSpellingValidateWord 326
TRichView 326
LoadDocX 326
TRichView 326
LoadDocXFromStream 327
TRichView 327
LoadField 614, 631
TDBRichView 614
TDBRichViewEdit 631
LoadFromFile 328
TRichView 328
LoadFromFileEx 328
TRichView 328
LoadFromRVST 801
TRVStyleTemplateCollection 801
LoadFromStream 329, 937
TRichView 329
TRVTableItemInfo 937
LoadFromStreamEx 329
TRichView 329
LoadFromStreamRVST 802
TRVStyleTemplateCollection 802
LoadHTML 330
TRichView 330
LoadHTMLFromStream 332
TRichView 332
LoadINI 771
TRVStyle 771
LoadMarkdown 332
TRichView 332
LoadMarkdownFromStream 333

LoadMarkdownFromStream 333
TRichView 333
LoadOptions 453
TRVMarkdownProperties 453
LoadReg 771
TRVStyle 771
LoadRTF 334
TRichView 334
LoadRTFFromStream 335
TRichView 335
LoadRVF 336
TRichView 336
LoadRVFFromStream 337
TRichView 337
LoadText 338
TRichView 338
LoadTextFromStream 339
TRichView 339
LoadTextFromStreamW 339
TRichView 339
LoadTextW 338
TRichView 338

- M -

MainDoc 1039
TRVReportHelperWithHeaderFooters 1039
MainRVStyle 756
TRVStyle 756
MakePreview 712
TRVPrint 712
MakeScaledPreview 713
TRVPrint 713
Margin 658
TRVChatCalloutProperties 658
MarginAbove 661
TRVChatDateProperties 661
MarginBelow 661
TRVChatDateProperties 661
Margins 705
TRVPrint 705
MarginsPen 892
TCustomRVPrintPreview 892
MarginsStroke 892
TCustomRVPrintPreview 892
MarkdownProperties 245
TRichView 245
MarkerAlignment 866
TRVListLevel 866
MarkerIndent 866
TRVListLevel 866
MarkMessageAsDeleted 653
TRVChat 653
MarkStylesInUse 339
TRichView 339
Mathematical expressions 193
MaxAutoQuoteLength 680
TRVChatQuoteProperties 680
MaxLength 246
TRichView 246
MaxPrintedItemNo 879
TCustomRVPrint 879
MaxTextWidth 246
TRichView 246
MaxWidth 658
TRVChatCalloutProperties 658
MB 680
TRVChatUITexts 680
Menu 601
TRVSmartPopupProperties 601
MergeCells 937
TRVTableItemInfo 937
MergeSelectedCells 938
TRVTableItemInfo 938
MessageFont 664
TRVChatFonts 664
MessageInfoColor 682
TRVChatUser 682
Messages 643
TRVChat 643
MessageStatusLine 643
TRVChat 643
MetafileCompatibility 880
TCustomRVPrint 880
MetafilePixelsPerInch 880
TCustomRVPrint 880
MinOppositeMargin 659
TRVChatCalloutProperties 659
MinPrintedItemNo 881
TCustomRVPrint 881
MinStatusLineHeight 677
TRVChatMessageStatusLine 677
MinTextWidth 246
TRichView 246
MinWidth 658, 971
TRVChatCalloutProperties 658
TRVLabelItemInfo 971
MirrorMargins 426, 706
TRVDocParameters 426
TRVPrint 706

Modified 487
TRichViewEdit 487
ModifiedProperties 824, 839
TFontInfo 824
TParaInfo 839
Modifying TRichView items 117
MoveCaret 542
TRichViewEdit 542
MoveMessage 653
TRVChat 653
MoveRows 938
TRVTableItemInfo 938
Msg 674, 676
TRVChatMessageExtraQuote 674
TRVChatMessageQuote 676
MustAddCopyOfImage 673
TRVChatMessageExtraImage 673

- N -

Name 682, 696, 845
TRVChatUser 682
TRVOfficeConverterInfo 696
TRVStyleTemplate 845
NameColor 682
TRVChatUser 682
NewImageIndex 679
TRVChatMessageStatusLine 679
Next 846, 897
TCustomRVPrintPreview 897
TRVStyleTemplate 846
NextId 846
TRVStyleTemplate 846
NextParaNo 840
TParaInfo 840
NextStyleNo 824
TFontInfo 824
NoMetafiles 881
TCustomRVPrint 881
NormalStyleTemplate 799
TRVStyleTemplateCollection 799
NotesSavedAsFootnotes 455
TRVMarkdownProperties 455
NoteText 247, 984
TCustomRVNoteItemInfo 984
TRichView 247
NoVScroll 873
TRVScroller 873
Numbered sequences 183
NumberType 980, 1009
TCustomRVPageNumberItemInfo 1009
TRVSeqItemInfo 980

- O -

Obtaining items of TRichView 116
OddColumnsColor 923
TRVTableItemInfo 923
OddRowsColor 923
TRVTableItemInfo 923
OffsetInCurItem 488
TRichViewEdit 488
OnAddStyle 377
TRichView 377
OnAfterApplyStyle 774
TRVStyle 774
OnAfterDrawImage 378
TRichView 378
OnAfterOleDrop 580
TRichViewEdit 580
OnAfterPrintImage 885
TCustomRVPrint 885
OnApplyStyle 774
TRVStyle 774
OnApplyStyleColor 775
TRVStyle 775
OnAssignImageFileName 379
TRichView 379
OnBeforeOleDrop 580
TRichViewEdit 580
OnCaretGetOut 581
TRichViewEdit 581
OnCaretMove 582
TRichViewEdit 582
OnCellEditing 948
TRVTableItemInfo 948
OnChange 582
TRichViewEdit 582
OnChanging 582
TRichViewEdit 582
OnCheckpointVisible 379
TRichView 379
OnCheckStickingItems 583
TRichViewEdit 583
OnClickDate 654
TRVChat 654
OnClickFile 654
TRVChat 654
OnClickImage 655
TRVChat 655

OnClickLink 655
TRVChat 655
OnClickUser 655
TRVChat 655
OnCMHintShow 380
TRichView 380
OnControlAction 380
TRichView 380
OnConverting 695
TRVOfficeConverter 695
OnCopy 381
TRichView 381
OnCurParaStyleChanged 583
TRichViewEdit 583
OnCurTextStyleChanged 584
TRichViewEdit 584
OnDrawBackground 949
TRVTableItemInfo 949
OnDrawBorder 949
TRVTableItemInfo 949
OnDrawCheckpoint 776, 885
TCustomRVPrint 885
TRVStyle 776
OnDrawCurrentLine 584
TRichViewEdit 584
OnDrawCustomCaret 585
TRichViewEdit 585
OnDrawHyperlink 886
TCustomRVPrint 886
OnDrawPageBreak 778
TRVStyle 778
OnDrawParaBack 780
TRVStyle 780
OnDrawStyleText 781
TRVStyle 781
OnDrawTextBack 783
TRVStyle 783
OnDropFile 586
TRichViewEdit 586
OnDropFiles 586
TRichViewEdit 586
OneLevelPreview 842
TRVListInfo 842
OnFormatting 717
TRVPrint 717
OnGetItemCursor 382
TRichView 382
OnGetPageNumber 728
TRVReportHelper 728
OnGetSpellingSuggestions 382
TRichView 382
OnHighlightedMessageChange 655
TRVChat 655
OnHScrolled 877
TRVScroller 877
OnHTMLSaveImage 383
TRichView 383
OnImportFile 385
TRichView 385
OnImportPicture 386
TRichView 386
OnItemAction 387
TRichView 387
OnItemHint 388
TRichView 388
OnItemResize 587
TRichViewEdit 587
OnItemTextEdit 588
TRichViewEdit 588
OnJump 390
TRichView 390
OnLoadCustomFormat 390
TRichView 390
OnLoadDocument 391
TRichView 391
OnMeasureCustomCaret 588
TRichViewEdit 588
OnNewDocument 391
TRichView 391
OnOleDragEnter 589
TRichViewEdit 589
OnOleDragLeave 589
TRichViewEdit 589
OnOleDragOver 590
TRichViewEdit 590
OnOleDrop 591
TRichViewEdit 591
OnPagePostpaint 718
TRVPrint 718
OnPagePrepaint 719
TRVPrint 719
OnPaint 392
TRichView 392
OnParaStyleConversion 592
TRichViewEdit 592
OnPaste 594
TRichViewEdit 594
OnPrintComponent 887
TCustomRVPrint 887
OnProgress 392

OnProgress 392
TRichView 392
OnReadField 393, 398
TCustomRichView 393, 398
OnReadHyperlink 395
TCustomRichView 395
OnRVDblClick 399
TRichView 399
OnRVFControlNeeded 400
TRichView 400
OnRVFImageListNeeded 400
TRichView 400
OnRVFPictureNeeded 401
TRichView 401
OnRVMouseDown 402
TRichView 402
OnRVMouseMove 403
TRichView 403
OnRVMouseUp 403
TRichView 403
OnRVRightClick 404
TRichView 404
OnSaveComponentToFile 405
TRichView 405
OnSaveCustomFormat 595
TRichViewEdit 595
OnSaveDocXExtra 407
TRichView 407
OnSaveHTMLExtra 409
TRichView 409
OnSaveImage2 410
TRichView 410
OnSaveItemToFile 411
TRichView 411
OnSaveParaToHTML 413
TRichView 413
OnSaveRTFExtra 414
TRichView 414
OnSelect 416
TRichView 416
OnSendingToPrinter 719
TRVPrint 719
OnSmartPopupClick 595
TRichViewEdit 595
OnSpellForm 737
TRVSpellChecker 737
OnSpellFormAction 738
TRVSpellChecker 738
OnSpellFormLocalize 739
TRVSpellChecker 739
OnSpellingCheck 416
TRichView 416
OnStyleConversion 595
TRichViewEdit 595
OnStyleHoverSensitive 784
TRVStyle 784
OnStyleTemplatesChange 417
TRichView 417
OnTextFound 418
TRichView 418
OnVScrolled 877
TRVScroller 877
OnWriteHyperlink 418
TRichView 418
OnWriteObjectProperties 420
TRichView 420
OnZoomChanged 899
TCustomRVPrintPreview 899
Opacity 852, 916, 958
TRVBackgroundRect 852
TRVTableCellData 958
TRVTableItemInfo 916
Options 247, 814, 833, 867, 916, 959
TCustomRVFontInfo 814
TCustomRVParaInfo 833
TRichView 247
TRVListLevel 867
TRVTableCellData 959
TRVTableItemInfo 916
Orientation 426
TRVDocParameters 426
OutlineLevel 835
TCustomRVParaInfo 835
Overview 89
Overview of item types 163, 164
Overview of tables 195

- P -

Padding 659, 680
TRVChatCalloutProperties 659
TRVChatQuoteProperties 680
PaddingHorizontal 661
TRVChatDateProperties 661
PaddingVertical 661
TRVChatDateProperties 661
Page counts 192
Page numbers 190
PageBorderColor 893
TCustomRVPrintPreview 893

PageBorderWidth 893
TCustomRVPrintPreview 893
PageBreakBefore 967
TRVTableRow 967
PageBreakColor 756
TRVStyle 756
PageBreaksBeforeItems 250
TRichView 250
PageHeight 426, 721
TRVDocParameters 426
TRVVirtualPrinterProperties 721
PageNo 893
TCustomRVPrintPreview 893
PageNoFromNumber 881
TCustomRVPrint 881
PagesCount 882
TCustomRVPrint 882
PageWidth 426, 721
TRVDocParameters 426
TRVVirtualPrinterProperties 721
ParaBackAroundContent 757
TRVStyle 757
Paragraphs in TRichView 100
ParaStyle 846
TRVStyleTemplate 846
ParaStyleMode 464
TRVRTFReaderProperties 464
ParaStyleNo 465
TRVRTFReaderProperties 465
ParaStyles 757
TRVStyle 757
ParaStyleTemplateId 815
TCustomRVFontInfo 815
Parent 847
TRVStyleTemplate 847
ParentId 847
TRVStyleTemplate 847
Paste 543
TRichViewEdit 543
PasteBitmap 544
TRichViewEdit 544
PasteGraphicFile 544
TRichViewEdit 544
PasteHTML 545
TRichViewEdit 545
PasteMarkdown 546
TRichViewEdit 546
PasteMetafile 546
TRichViewEdit 546
PasteRTF 547
TRichViewEdit 547
PasteRVF 547
TRichViewEdit 547
PasteText 548
TRichViewEdit 548
PasteTextA 548
TRichViewEdit 548
PasteTextW 548
TRichViewEdit 548
PasteURL 549
TRichViewEdit 549
Path 696
TRVOfficeConverterInfo 696
Picture 683, 868
TRVChatUser 683
TRVListLevel 868
PictureArchive 663
TRVChatDateProperties 663
PictureAudio 663
TRVChatDateProperties 663
PictureDocument 663
TRVChatDateProperties 663
PictureDownload 663
TRVChatDateProperties 663
PictureImage 663
TRVChatDateProperties 663
Pictures 169
PictureVideo 663
TRVChatDateProperties 663
PixelsPerInch 786
TFontInfos 786
PixelsToUnits 769
TRVStyle 769
Popup 601
TRVSmartPopupProperties 601
Position 601, 685, 857
TRVChatUserPictureProperties 685
TRVSmartPopupProperties 601
TRVTabInfo 857
PositionInText 1001
TRVBoxPosition 1001
Prev 898
TCustomRVPrintPreview 898
Preview100PercentHeight 706
TRVPrint 706
Preview100PercentWidth 706
TRVPrint 706
PreviewCorrection 882
TCustomRVPrint 882
PreviewMode 691

PreviewMode 691
TRVOfficeConverter 691
Print 713
TRVPrint 713
PrintableAreaPen 893
TCustomRVPrintPreview 893
PrintableAreaStroke 893
TCustomRVPrintPreview 893
PrintOptions 919
TRVTableItemInfo 919
PrintPages 714
TRVPrint 714
Protection 815
TCustomRVFontInfo 815
ProtectTextStyleNo 972
TRVLabelItemInfo 972
PRVPDFMetadata 1084
PRVScreenAndDevice 1090

- Q -

QuickAccess 847
TRVStyleTemplate 847
Quote 643, 667
TRVChat 643
TRVChatMessage 667
QuoteFillColor 683
TRVChatUser 683
QuoteFont 664
TRVChatFonts 664
QuoteType 676, 680
TRVChatMessageQuote 676

- R -

ReadBackground 434
TRVHTMLReaderProperties 434
ReadDocParameters 435, 465
TRVHTMLReaderProperties 435
TRVRTFReaderProperties 465
ReadImageIndex 679
TRVChatMessageStatusLine 679
ReadNoteNumberingType 465
TRVRTFReaderProperties 465
ReadOnly 488, 624
TDBRichViewEdit 624
TRichViewEdit 488
Redo 549
TRichViewEdit 549
RedoAction 549
TRichViewEdit 549
RedoName 550
TRichViewEdit 550
References to parent endnotes and footnotes 190
Reformat 339
TRichView 339
RefreshListMarkers 340
TRichView 340
RefreshSequences 340
TRichView 340
RegisterAsService 735
TRVSpellChecker 735
RegisterEditor 736
TRVSpellChecker 736
RelativeToCell 1002
TRVBoxPosition 1002
RemoveCheckpoint 340
TRichView 340
RemoveCheckpointAtCaret 550
TRichViewEdit 550
RemoveCheckpointEd 550
TRichViewEdit 550
RemoveCurrentCheckpoint 551
TRichViewEdit 551
RemoveCurrentPageBreak 551
TRichViewEdit 551
RemoveInternalLeading 972
TRVLabelItemInfo 972
RemoveLists 552
TRichViewEdit 552
RemoveSpaceBelow 972
TRVLabelItemInfo 972
ReplaceDocumentEd 985
TCustomRVNoteItemInfo 985
Reset 429, 981, 1041
TRVDocParameters 429
TRVReportHelperWithHeaderFooters 1041
TRVSeqItemInfo 981
ResetAnimation 341
TRichView 341
ResetLayout 429
TRVDocParameters 429
ResetNameCounter 802
TRVStyleTemplateCollection 802
ResizeControl 552
TRichViewEdit 552
ResizeCurrentControl 552
TRichViewEdit 552
RichView 36, 724
TRVReportHelper 724

RichView Format (RVF) overview 129
RichViewAllowCopyTableCells 1110
RichViewAlternativePicPrint 1110
RichViewApostropheInWord 1111
RichViewAutoAssignNormalStyleTemplate 1111
RichViewCompareStyleNames 1111
RichViewCSSUseSystemColors 1112
RichViewCtrlUpDownScroll 1112
RichViewDoNotCheckRVFStyleRefs 1112
RichViewDoNotMergeNumbering 1113
RichViewEditCaretHeightExtra 1108
RichViewEditCaretPosition 1108
RichViewEditCaretWidth 1108
RichViewEditCaretWidthMode 1108
RichViewEditCustomCaretSize 1108
RichViewEditDefaultProportionalResize 1113
RichViewEditEnterAllowsEmptyMarkeredLines 1113
RichViewEditMaxCaretHeight 1108
RichViewEditMinCaretHeight 1108
RichViewJustifyBeforeLineBreak 1114
RichViewMaxPictureCount 1114
RichViewMaxThumbnailCount 1114
RichViewPixelsPerInch 721, 1115
RichViewRemoveNbSpAfterImport 1115
RichViewResetStandardFlag 1115
RichViewSaveDivInHTMLEx 1107
RichViewSaveHyperlinksInDocXAsFields 1116
RichViewSavePageBreaksInText 1107
RichViewSavePInHTML 1107
RichViewShowGhostSpaces 1116
RichViewStringFormatMethod 1116
RichViewTableAutoAddRow 1117
RichViewTableDefaultRTFAutofit 1117
RichViewToLinear 1047
RichViewUnicodeInput 1118
RightIndent 835
  TCustomRVParaInfo 835
RightMargin 251, 426
  TRichView 251
  TRVDocParameters 426
Rotation 961
  TRVTableCellData 961
RowBandSize 913
  TRVTableItemInfo 913
RowCount 920
  TRVTableItemInfo 920
Rows 920
  TRVTableItemInfo 920
RowSpan 962
  TRVTableCellData 962
RTFErrorCode 466
  TRVRTFReaderProperties 466
RTFOptions 252
  TRichView 252
RTFormatName 1104
RTFReadProperties 253
  TRichView 253
Rules of tables 199
RV_AfterImportGraphic 1136
RV_CP_DEFAULT 1118
RV_CreateGraphics 1136
RV_EMUToUnits 1044, 1049
RV_GetPrinterDC 1050
RV_RegisterHTMLGraphicFormat 1136
RV_RegisterPngGraphic 1136
RV_TestFileUnicode 1054
RV_TestStreamUnicode 1054
RV_TestStringUnicode 1054
RV_TwipsToUnits 1044, 1049
RV_UnitsPerInch 1044, 1049
RV_UnitsToEMU 1044, 1049
RV_UnitsToPixels 1044, 1049
RV_UnitsToTwips 1044, 1049
RV_UnitsToUnits 1044, 1049
RVAllFontInfoProperties 1075
RVAllParaInfoProperties 1083
RVAlwaysShowPlaceholders 1119
RVApplyMathProperties 1048
RVCharsPerLineBreak 1047
RVChatMessageClass 643
RVChatUserClass 644
rvclNone 812
RVControlsPainter 1119
RVData 254
  TRichView 254
RVDefaultLoadProperties 1119
rvdoALL 1065
rveipcAlignment 1119, 1124
rveipcBreakColor 1119
rveipcBreakStyle 1119
rveipcBreakWidth 1119
rveipcCursor 1119, 1124
rveipcDisplayInline 1119
rveipcFontSizeDouble 1119
rveipcHotImageIndex 1119
rveipcMinWidth 1119, 1124
rveipcPercentWidth 1119
rveipcProtectTextStyleNo 1119, 1124

rveipcRemoveInternalLeading 1119, 1124
rveipcSeqNumberType 1119, 1124
rveipcSeqReset 1119, 1124
rveipcSeqStartFrom 1119, 1124
rveipcTextColor 1119
RVEMPTYTAG 1124
RVF 129, 148, 237
RVF Specification 148
RVFOptions 254
  TRichView 254
RVFormatName 1104
RVFParaStylesReadMode 258
  TRichView 258
RVFTextStylesReadMode 258
  TRichView 258
RVFWarnings 258
  TRichView 258
RVGetFirstEndnote 1052
RVGetFirstFootnote 1052
RVGetFirstSidenote 1052
RVGetLinearCaretPos 1047
RVGetMathProperties 1048
RVGetNextEndnote 1052
RVGetNextFootnote 1052
RVGetNextSidenote 1052
RVGetNoteTextStyleNo 1052
RVGetSelection 1047
RVGetSelectionEx 1047
RVGetSystemColor 1125
RVGetTextLength 1047
RVGetTextRange 1047
RVGraphicHandler 1125
RVIsCustomURL 1126
RVIsEmail function 1053
RVIsURL function 1053
RVItemFactory function 1054
RVMathEquationManager function 1054
RVNonTextCharacter 1047
RVOpenURL procedure 1054
RVParagraphMarks 1126
RVPrint 638
  TRVPrintPreview 638
rvsBreak 1127
rvsBullet 1127
rvsComponent 1127
rvsDefStyle 1127
rvsEndnote 1127
RVSetLinearCaretPos 1047
RVSetMathProperties 1048
RVSetSelection 1047
RVSetSelectionEx 1047
rvsFootnote 1127
rvsHeading 1127
rvsHotPicture 1127
rvsHotspot 1127
rvsJump1 1127
rvsJump2 1127
rvsKeyword 1127
rvsLabel 1127
rvsListMarker 1127
rvsMathEquation 1127
rvsNormal 1127
rvsNoteReference 1127
rvsPicture 1127
rvsSequence 1127
rvsSubheading 1127
rvsTab 1127
rvsTable 1127
RVStyle.TRVColorMode 1063
RVThumbnailMaker 1129
RVU_AnsiToUnicode 1054
RVU_AnsiToUTF8 1054
RVU_UnicodeToAnsi 1054
RVUnitsToUnits 769
  TRVStyle 769
RVVisibleSpecialCharacters 1129

- S -

SaveCSS 772
  TRVStyle 772
SaveCSSToStream 772
  TRVStyle 772
SaveDocX 341
  TRichView 341
SaveDocXToStream 342
  TRichView 342
SaveHeaderAndFooter 445
  TRVHTMLSaveProperties 445
SaveHiddenText 456
  TRVMarkdownProperties 456
SaveHTML 342, 343
  TRichView 342, 343
SaveHTMLEx 345
  TRichView 345
SaveHTMLToStream 346, 347
  TRichView 346, 347
SaveHTMLToStreamEx 348

SaveHTMLToStreamEx 348
TRichView 348
SaveINI 773
TRVStyle 773
SaveMarkdown 348
TRichView 348
SaveMarkdownToStream 349
TRichView 349
SaveMathMLInHTML 1021
SaveOMMLInDocX 1021
SaveOptions 456
TRVMarkdownProperties 456
SavePDF 715, 727
TRVPrint 715
TRVReportHelper 727
SavePicture 349
TRichView 349
SaveReg 773
TRVStyle 773
SaveRowsToStream 939
TRVTableItemInfo 939
SaveRTF 350
TRichView 350
SaveRTFToStream 351
TRichView 351
SaveRVF 351
TRichView 351
SaveRVFToStream 352
TRichView 352
SaveText 352
TRichView 352
SaveTextToStream 353
TRichView 353
SaveTextToStreamW 353
TRichView 353
SaveTextW 352
TRichView 352
SaveToRVST 801
TRVStyleTemplateCollection 801
SaveToStream 939
TRVTableItemInfo 939
SaveToStreamRVST 802
TRVStyleTemplateCollection 802
ScalePictures 663
TRVChatDateProperties 663
Scrolling in TRichView 110
ScrollTo 875
TRVScroller 875
ScrollToMessage 654
TRVChat 654
Searching and replacing in TRichView and TRichViewEdit 115
SearchText 354, 553
TRichView 354
TRichViewEdit 553
SearchTextA 354, 553
TRichView 354
TRichViewEdit 553
SearchTextW 354, 553
TRichView 354
TRichViewEdit 553
SelColor 751
TRVStyle 751
Select 939
TRVTableItemInfo 939
SelectAll 355
TRichView 355
SelectCols 940
TRVTableItemInfo 940
SelectControl 355
TRichView 355
SelectCurrentLine 554
TRichViewEdit 554
SelectCurrentWord 555
TRichViewEdit 555
Selecting parts of TRichView document 112
Selection in Table 201
SelectionExists 356
TRichView 356
SelectionHandleKind 757
TRVStyle 757
SelectionHandlesVisible 260
TRichView 260
SelectionMode 758
TRVStyle 758
SelectionStyle 759
TRVStyle 759
SelectRows 940
TRVTableItemInfo 940
SelectWordAt 356
TRichView 356
SelOpacity 751
TRVStyle 751
SelTextColor 751
TRVStyle 751
Semitransparent Objects in TRichView 126
SentImageIndex 679
TRVChatMessageStatusLine 679
SeqName 981
TRVSeqItemInfo 981

SetAddParagraphMode 357
TRichView 357

SetBackgroundImageEd 556
TRichViewEdit 556

SetBackgroundProperty 940
TRVTableItemInfo 940

SetBackgroundPropertyEd 557
TRichViewEdit 557

SetBreakInfo 358
TRichView 358

SetBreakInfoEd 557
TRichViewEdit 557

SetBulletInfo 359
TRichView 359

SetBulletInfoEd 558
TRichViewEdit 558

SetButtonState 602
TRVSmartPopupProperties 602

SetCellBackgroundImage 941
TRVTableItemInfo 941

SetCellBackgroundImageFileName 941
TRVTableItemInfo 941

SetCellBackgroundProperty 941
TRVTableItemInfo 941

SetCellBackgroundStyle 942
TRVTableItemInfo 942

SetCellBestHeight 942
TRVTableItemInfo 942

SetCellBestWidth 942
TRVTableItemInfo 942

SetCellBorderColor 942
TRVTableItemInfo 942

SetCellBorderLightColor 943
TRVTableItemInfo 943

SetCellColor 943
TRVTableItemInfo 943

SetCellHint 943
TRVTableItemInfo 943

SetCellOpacity 943
TRVTableItemInfo 943

SetCellOptions 944
TRVTableItemInfo 944

SetCellRotation 944
TRVTableItemInfo 944

SetCellTag 944
TRVTableItemInfo 944

SetCellVAlign 945
TRVTableItemInfo 945

SetCellVisibleBorders 945
TRVTableItemInfo 945

SetCheckpointInfo 360
TRichView 360

SetCheckpointInfoEd 560
TRichViewEdit 560

SetControlInfo 361
TRichView 361

SetControlInfoEd 560
TRichViewEdit 560

SetCurrentBreakInfo 561
TRichViewEdit 561

SetCurrentBulletInfo 562
TRichViewEdit 562

SetCurrentCheckpointInfo 563
TRichViewEdit 563

SetCurrentControlInfo 564
TRichViewEdit 564

SetCurrentHotspotInfo 565
TRichViewEdit 565

SetCurrentItemExtraIntProperty 566
TRichViewEdit 566

SetCurrentItemExtraIntPropertyEx 566
TRichViewEdit 566

SetCurrentItemExtraStrProperty 567
TRichViewEdit 567

SetCurrentItemExtraStrPropertyEx 567
TRichViewEdit 567

SetCurrentItemText 568
TRichViewEdit 568

SetCurrentItemTextA 568
TRichViewEdit 568

SetCurrentItemTextW 568
TRichViewEdit 568

SetCurrentItemVAlign 569
TRichViewEdit 569

SetCurrentPictureInfo 569
TRichViewEdit 569

SetCurrentTag 570
TRichViewEdit 570

SetDefaultFontName 664
TRVChatFonts 664

SetFloatPropertyEd 555
TRichViewEdit 555

SetFontName 664
TRVChatFonts 664

SetFooter 362, 715
TRichView 362
TRVPrint 715

SetHeader 363, 716
TRichView 363
TRVPrint 716

SetHotspotInfo 364
TRichView 364
SetHotspotInfoEd 570
TRichViewEdit 570
SetIntPropertyEd 555
TRichViewEdit 555
SetItemExtraIntProperty 366
TRichView 366
SetItemExtraIntPropertyEd 572
TRichViewEdit 572
SetItemExtraIntPropertyEx 366
TRichView 366
SetItemExtraIntPropertyExEd 572
TRichViewEdit 572
SetItemExtraStrProperty 367
TRichView 367
SetItemExtraStrPropertyEd 573
TRichViewEdit 573
SetItemExtraStrPropertyEx 367
TRichView 367
SetItemExtraStrPropertyExEd 573
TRichViewEdit 573
SetItemTag 367
TRichView 367
SetItemTagEd 574
TRichViewEdit 574
SetItemText 368
TRichView 368
SetItemTextA 368
TRichView 368
SetItemTextEd 574
TRichViewEdit 574
SetItemTextEdA 574
TRichViewEdit 574
SetItemTextEdW 574
TRichViewEdit 574
SetItemTextW 368
TRichView 368
SetItemVAlign 369
TRichView 369
SetItemVAlignEd 575
TRichViewEdit 575
SetListMarkerInfo 369
TRichView 369
SetPictureInfo 370
TRichView 370
SetPictureInfoEd 576
TRichViewEdit 576
SetRowKeepTogether 945
TRVTableItemInfo 945
SetRowPageBreakBefore 945
TRVTableItemInfo 945
SetRowVAlign 945
TRVTableItemInfo 945
SetSelectionBounds 372
TRichView 372
SetStrPropertyEd 555
TRichViewEdit 555
SetTableVisibleBorders 946
TRVTableItemInfo 946
SetUndoGroupMode 577
TRichViewEdit 577
SetZoom 898
TCustomRVPrintPreview 898
ShadowColor 894
TCustomRVPrintPreview 894
ShadowOpacity 659
TRVChatCalloutProperties 659
ShadowWidth 894
TCustomRVPrintPreview 894
ShortCut 601
TRVSmartPopupProperties 601
ShowNameInMessages 683
TRVChatUser 683
ShowPicture 683
TRVChatUser 683
SidenoteNumbering 746
TRVStyle 746
Sidenotes 186
SingleClick 260
TRichView 260
SingleSymbols 815
TCustomRVFontInfo 815
Size 685, 817
TCustomRVFontInfo 817
TRVChatUserPictureProperties 685
SizeDouble 817
TCustomRVFontInfo 817
Skia 163
Skia4Delphi 163
SkipHiddenText 435, 466
TRVHTMLReaderProperties 435
TRVRTFReaderProperties 466
Smart Popups 598
SmartPopupProperties 489
TRichViewEdit 489
SmartPopupVisible 489
TRichViewEdit 489
Smooth image scaling in TRichView 125
SoftPageBreakColor 761

SoftPageBreakColor 761
TRVStyle 761
SpaceAfter 835
TCustomRVParaInfo 835
SpaceBefore 836
TCustomRVParaInfo 836
SpacesInTab 761
TRVStyle 761
Spacing 659
TRVChatCalloutProperties 659
SpecialCharactersColor 762
TRVStyle 762
SpellFormStyle 732
TRVSpellChecker 732
SplitSelectedCellsHorizontally 946
TRVTableItemInfo 946
SplitSelectedCellsVertically 946
TRVTableItemInfo 946
Standard 804
TCustomRVInfo 804
StandardPixelsToUnits 769
TRVStyle 769
StartAnimation 374
TRichView 374
StartAt 882
TCustomRVPrint 882
StartFrom 868, 981
TRVListLevel 868
TRVSeqItemInfo 981
StartLiveSpelling 374, 736
TRichView 374
TRVSpellChecker 736
State 667
TRVChatMessage 667
StatusLineContent 678
TRVChatMessageStatusLine 678
StatusLineLocation 678
TRVChatMessageStatusLine 678
StopAnimation 375
TRichView 375
StoreSearchResult 375
TRichView 375
Stream 691
TRVOfficeConverter 691
Style 260, 817, 854
TCustomRVFontInfo 817
TRichView 260
TRVBorder 854
style templates 839
StyleEx 818
TCustomRVFontInfo 818
StyleName 805
TCustomRVInfo 805
Styles and Style-Templates in TRichView 103
StyleTemplateId 805
TCustomRVFontOrParaInfo 805
StyleTemplateInsertMode 260
TRichView 260
StyleTemplates 762
TRVStyle 762
SubSuperScriptType 818
TCustomRVFontInfo 818
Suggest 737
TRVSpellChecker 737
SupportsFeatures 737
TRVSpellChecker 737

- T -

Table Cells 198
Table Operations 205
Table Resizing 211
TableCellWidthPix 447
TRVMarkdownDefaultItemProperties 447
Tables 179
Tables and Compatibility 196
Tables in RichView 196
Tables in TRichView 195
TabNavigation 262
TRichView 262
Tabs 836
TCustomRVParaInfo 836
Tabulators 168
Tag 963
TRVTableCellData 963
Tags of TRichView items 96
TargetPixelsPerInch 723
TRVReportHelper 723
TBackgroundStyle 229
TCheckpointData 1055
TConvertingEvent 695
TCPEventKind 233
TCustomRichView 216, 401
TCustomRichView.AssignSoftPageBreaks 284
TCustomRichView.GetItemAt 305
TCustomRichView.GetItemExtraStrProperty 308
TCustomRichView.GetItemExtraStrPropertyEx 308
TCustomRichView.OnReadField 393, 398
TCustomRichView.OnReadHyperlink 395

TCustomRichViewEdit 469, 1048
TCustomRichViewEdit.ActualCurTextStyleNo 480
TCustomRVControl 869
TCustomRVData 1014
TCustomRVData.Edit 1016
TCustomRVData.GetRVData 1017
TCustomRVData.GetSourceRVData 1017
TCustomRVFieldItemInfo 1009
TCustomRVFontInfo 806
TCustomRVFontInfo.Assign 820
TCustomRVFontInfo.AssignTo 820
TCustomRVFontInfo.BackColor 808
TCustomRVFontInfo.BiDiMode 808
TCustomRVFontInfo.CharScale 808
TCustomRVFontInfo.Charset 809
TCustomRVFontInfo.CharSpacing 809
TCustomRVFontInfo.Color 810
TCustomRVFontInfo.Create 821
TCustomRVFontInfo.EmptyWidth 810
TCustomRVFontInfo.FontName 811
TCustomRVFontInfo.HoverBackColor 811
TCustomRVFontInfo.HoverColor 812
TCustomRVFontInfo.HoverEffects 812
TCustomRVFontInfo.HoverUnderlineColor 813
TCustomRVFontInfo.IsEqual 821
TCustomRVFontInfo.JumpCursor 813
TCustomRVFontInfo.Options 814
TCustomRVFontInfo.ParaStyleTemplateId 815
TCustomRVFontInfo.Protection 815
TCustomRVFontInfo.SingleSymbols 815
TCustomRVFontInfo.Size 817
TCustomRVFontInfo.SizeDouble 817
TCustomRVFontInfo.Style 817
TCustomRVFontInfo.StyleEx 818
TCustomRVFontInfo.SubSuperScriptType 818
TCustomRVFontInfo.UnderlineColor 818
TCustomRVFontInfo.UnderlineType 819
TCustomRVFontInfo.VShift 820
TCustomRVFontOrParaInfo.StyleTemplateId 805
TCustomRVFormattedData 1014
TCustomRVFormattedData.GetOriginEx 1018
TCustomRVInfo 803
TCustomRVInfo.Standard 804
TCustomRVInfo.StyleName 805
TCustomRVIsURLFunction 1126
TCustomRVItemInfo 899
TCustomRVNoteItemInfo 899, 982
TCustomRVNoteItemInfo.CreateEx 984
TCustomRVNoteItemInfo.Document 984
TCustomRVNoteItemInfo.NoteText 984
TCustomRVNoteItemInfo.ReplaceDocumentEd 985
TCustomRVPageNumberItemInfo 1009
TCustomRVPageNumberItemInfo.NumberType 1009
TCustomRVParaInfo 828
TCustomRVParaInfo.Alignment 829
TCustomRVParaInfo.Assign 836
TCustomRVParaInfo.Background 830
TCustomRVParaInfo.BiDiMode 831
TCustomRVParaInfo.Border 831
TCustomRVParaInfo.Create 836
TCustomRVParaInfo.FirstIndent 831
TCustomRVParaInfo.IsEqual 837
TCustomRVParaInfo.LastLineAlignment 829
TCustomRVParaInfo.LeftIndent 832
TCustomRVParaInfo.LineSpacing 832
TCustomRVParaInfo.LineSpacingType 833
TCustomRVParaInfo.Options 833
TCustomRVParaInfo.OutlineLevel 835
TCustomRVParaInfo.RightIndent 835
TCustomRVParaInfo.SpaceAfter 835
TCustomRVParaInfo.SpaceBefore 836
TCustomRVParaInfo.Tabs 836
TCustomRVPrint 877
TCustomRVPrint.CanSavePDF 883
TCustomRVPrint.Clear 883
TCustomRVPrint.ColorMode 878
TCustomRVPrint.DarkMode 878
TCustomRVPrint.EndAt 878
TCustomRVPrint.FirstPageNo 879
TCustomRVPrint.GetFirstItemOnPage 883
TCustomRVPrint.GetPageNo 883
TCustomRVPrint.IgnorePageBreaks 879
TCustomRVPrint.LastPageNo 879
TCustomRVPrint.MaxPrintedItemNo 879
TCustomRVPrint.MetafileCompatibility 880
TCustomRVPrint.MetafilePixelsPerInch 880
TCustomRVPrint.MinPrintedItemNo 881
TCustomRVPrint.NoMetafiles 881
TCustomRVPrint.OnAfterPrintImage 885
TCustomRVPrint.OnDrawCheckpoint 885
TCustomRVPrint.OnDrawHyperlink 886
TCustomRVPrint.OnPrintComponent 887
TCustomRVPrint.PageNoFromNumber 881
TCustomRVPrint.PagesCount 882
TCustomRVPrint.PreviewCorrection 882
TCustomRVPrint.StartAt 882
TCustomRVPrint.TransparentBackground 882
TCustomRVPrint.UpdatePaletteInfo 884

TCustomRVPrintPreview 888
TCustomRVPrintPreview.ClickMode 891
TCustomRVPrintPreview.Color 891
TCustomRVPrintPreview.Create 897
TCustomRVPrintPreview.DarkModeUI 892
TCustomRVPrintPreview.Fill 891
TCustomRVPrintPreview.First 897
TCustomRVPrintPreview.Last 897
TCustomRVPrintPreview.MarginsPen 892
TCustomRVPrintPreview.MarginsStroke 892
TCustomRVPrintPreview.Next 897
TCustomRVPrintPreview.OnZoomChanged 899
TCustomRVPrintPreview.PageBorderColor 893
TCustomRVPrintPreview.PageBorderWidth 893
TCustomRVPrintPreview.PageNo 893
TCustomRVPrintPreview.Prev 898
TCustomRVPrintPreview.PrintableAreaPen 893
TCustomRVPrintPreview.PrintableAreaStroke 893
TCustomRVPrintPreview.SetZoom 898
TCustomRVPrintPreview.ShadowColor 894
TCustomRVPrintPreview.ShadowWidth 894
TCustomRVPrintPreview.ZoomInCursor 895
TCustomRVPrintPreview.ZoomMode 895
TCustomRVPrintPreview.ZoomOutCursor 896
TCustomRVPrintPreview.ZoomPercent 896
TDBRichView 604
TDBRichView.AllowMarkdown 608
TDBRichView.AutoDeleteUnusedStyles 608
TDBRichView.AutoDisplay 609
TDBRichView.CodePage 609
TDBRichView.Create 614
TDBRichView.DataField 610
TDBRichView.DataSource 610
TDBRichView.Destroy 614
TDBRichView.Field 610
TDBRichView.LoadField 614
TDBRichViewEdit 616
  Inserting controls what can modify themselves in DBRichViewEdit 634
  Using viewer-style methods in DBRichViewEdit 634
TDBRichViewEdit.AllowMarkdown 621
TDBRichViewEdit.AutoDeleteUnusedStyles 621
TDBRichViewEdit.AutoDisplay 622
TDBRichViewEdit.Change 631
TDBRichViewEdit.CodePage 622
TDBRichViewEdit.Create 631
TDBRichViewEdit.DataField 622
TDBRichViewEdit.DataSource 623
TDBRichViewEdit.Destroy 631
TDBRichViewEdit.Field 623
TDBRichViewEdit.FieldFormat 623
TDBRichViewEdit.IgnoreEscape 624
TDBRichViewEdit.LoadField 631
TDBRichViewEdit.ReadOnly 624
TDBRichViewEdit.TRVDocumentProperty 432
Text 667, 674, 676, 972, 977
  TRVChatMessage 667
  TRVChatMessageExtraQuote 674
  TRVChatMessageQuote 676
  TRVLabelItemInfo 972
  TRVMathItemInfo 977
Text boxes 188
Text Items 167
TextBackgroundKind 762
  TRVStyle 762
TextColor 977
  TRVMathItemInfo 977
TextColSeparator 920
  TRVTableItemInfo 920
TextEngine 763
  TRVStyle 763
TextRowSeparator 921
  TRVTableItemInfo 921
Texts 643
  TRVChat 643
TextStyle 847
  TRVStyleTemplate 847
TextStyleMode 466
  TRVRTFReaderProperties 466
TextStyleNo 467, 973
  TRVLabelItemInfo 973
  TRVRTFReaderProperties 467
TextStyles 764
  TRVStyle 764
TFontInfo 821
TFontInfo.Assign 825
TFontInfo.Create 825
TFontInfo.Draw 825
TFontInfo.IsEqual 827
TFontInfo.Jump 823
TFontInfo.ModifiedProperties 824
TFontInfo.NextStyleNo 824
TFontInfo.Unicode 824
TFontInfos 785
TFontInfos.Add 787
TFontInfos.AddFont 787
TFontInfos.AddFontEx 788
TFontInfos.AssignTo 788

TFontInfos.Create 788
TFontInfos.FindStyleWithCharset 788
TFontInfos.FindStyleWithColor 788
TFontInfos.FindStyleWithFont 789
TFontInfos.FindStyleWithFontName 789
TFontInfos.FindStyleWithFontSize 790
TFontInfos.FindStyleWithFontStyle 790
TFontInfos.FindSuchStyle 791
TFontInfos.FindSuchStyleEx 791
TFontInfos.InvalidItem 786
TFontInfos.Items 786
TFontInfos.PixelsPerInch 786
TimeFont 664
  TRVChatFonts 664
Title 427
  TRVDocParameters 427
TitlePage 427, 706
  TRVDocParameters 427
  TRVPrint 706
TJumpEvent 390
TopLevelEditor 489
  TRichViewEdit 489
TopMargin 262, 427
  TRichView 262
  TRVDocParameters 427
TParaInfo 837
TParaInfo.Assign 840
TParaInfo.Create 840
TParaInfo.DefStyleNo 839
TParaInfo.IsEqual 841
TParaInfo.ModifiedProperties 839
TParaInfo.NextParaNo 840
TParaInfos 791
TParaInfos.Add 793
TParaInfos.AssignTo 793
TParaInfos.Create 794
TParaInfos.FindStyleWithAlignment 794
TParaInfos.FindSuchStyle 794
TParaInfos.FindSuchStyleEx 794
TParaInfos.InvalidItem 792
TParaInfos.Items 792
Tracking 873
  TRVScroller 873
TransparentBackground 882
  TCustomRVPrint 882
TRichView 216, 786
  adding items 107
  Component Editor 224
  live spelling check 138
  Loading UTF-8 files 468
  modifying items 117
  New after version 23 38
  New in v1.2 87
  New in v1.3 86
  New in v1.4 85
  New in v1.5 82
  New in v1.6 80
  New in v1.7 77
  New in v1.8 75
  New in v1.9 72
  New in v10 68
  New in v11 65
  New in v12 64
  New in v13 62
  New in v14 58
  New in v15 56
  New in v16 54
  New in v17 51
  New in v18 48
  New in v19 47
  New in v20 45
  New in version 21 43
  New in version 22 42
  New in version 23 40
  obtaining items 116
  selecting parts of document 112
  Version history 38
TRichView Clipboard functions overview 113
TRichView constants and variables
  (List) 1102
  CFRV_HTML 1104
  CFRV_RTF 1104
  CFRV_RVF 1104
  CFRV_URL 1104
  color constants 1105
  DefaultRVFPixelsPerInch 1109
  HTMLFormatName 1104
  pen style constants 1110
  RichViewAllowCopyTableCells 1110
  RichViewAlternativePicPrint 1110
  RichViewAutoAssignNormalStyleTemplate 1111
  RichViewCompareStyleNames 1111
  RichViewCSSUseSystemColors 1112
  RichViewCtrlUpDownScroll 1112
  RichViewDoNotCheckRVFStyleRefs 1112
  RichViewDoNotMergeNumbering 1113
  RichViewEditDefaultProportionalResize 1113
  RichViewEditEnterAllowsEmptyMarkeredLines
  1113
  RichViewJustifyBeforeLineBreak 1114

TRichView constants and variables
RichViewMaxPictureCount 1114
RichViewMaxThumbnailCount 1114
RichViewPixelsPerInch 1115
RichViewRemoveNbSpAfterImport 1115
RichViewResetStandardFlag 1115
RichViewSaveDivInHTMLEx 1107
RichViewSaveHyperlinksInDocXAsFields 1116
RichViewSavePageBreaksInText 1107
RichViewSavePInHTML 1107
RichViewShowGhostSpaces 1116
RichViewStringFormatMethod 1116
RichViewTableAutoAddRow 1117
RichViewTableDefaultRTFAutofit 1117
RichViewUnicodeInput 1118
RTFormatName 1104
RVAlwaysShowPlaceholders 1119
rveipcAlignment 1119, 1124
rveipcBreakColor 1119
rveipcBreakStyle 1119
rveipcBreakWidth 1119
rveipcCursor 1119, 1124
rveipcHotImageIndex 1119
rveipcImageIndex 1119
rveipcMinWidth 1119, 1124
rveipcPercentWidth 1119
rveipcProtectTextStyleNo 1119, 1124
rveipcRemoveInternalLeading 1119, 1124
rveipcSeqNumberType 1119, 1124
rveipcSeqReset 1119, 1124
rveipcSeqStartFrom 1119, 1124
RVFormatName 1104
RVGetSystemColor 1125
RVIsCustomURL 1126
RVParagraphMarks 1126
rvsBreak 1127
rvsBullet 1127
rvsComponent 1127
rvsDefStyle 1127
rvsEndnote 1127
rvsFootnote 1127
rvsHeading 1127
rvsHotPicture 1127
rvsHotspot 1127
rvsJump1 1127
rvsJump2 1127
rvsKeyword 1127
rvsLabel 1127
rvsListMarker 1127
rvsNormal 1127
rvsNoteReference 1127
rvsPicture 1127
rvsSequence 1127
rvsSubheading 1127
rvsTab 1127
rvsTable 1127
RVVisibleSpecialCharacters 1129
URLFormatName 1104
TRichView hypertext 97
TRichView item types 90, 163, 899
TRichView items' "checkpoints" overview 92
TRichView items' "tags" overview 96
TRichView methods for saving and loading 127
TRichView overview 36, 89
TRichView paragraphs 100
TRichView procedures and functions
DrawButton 1043
DrawControl 1043
DrawEdit 1043
DrawMemo 1043
DrawPanel 1043
GetRVESearchOptions 1049
GetRVSearchOptions 1049
RV_GetPrinterDC 1050
RV_TestFileUnicode 1054
RV_TestStreamUnicode 1054
RV_TestStringUnicode 1054
RVGetFirstEndnote 1052
RVGetFirstFootnote 1052
RVGetFirstSidenote 1052
RVGetNextEndnote 1052
RVGetNextFootnote 1052
RVGetNextSidenote 1052
RVGetNoteTextStyleNo 1052
RVIsEmail 1053
RVIsURL 1053
RVItemFactory 1054
RVMathEquationManager 1054
RVOpenURL 1054
RVU_AnsiToUnicode 1054
RVU_AnsiToUTF8 1054
RVU_GetRawUnicode 1054
RVU_RawUnicodeToWideString 1054
RVU_UnicodeToAnsi 1054
TRichView scrolling 110
TRichView.Add 277
TRichView.AddBreak 269
TRichView.AddBreakEx 269
TRichView.AddBreakExTag 269
TRichView.AddBreakTag 269
TRichView.AddBullet 270

TRichView.AddBulletEx 270
TRichView.AddBulletExTag 270
TRichView.AddCheckpoint 271
TRichView.AddCheckpointTag 271
TRichView.AddControl 272
TRichView.AddControlEx 272
TRichView.AddControlExTag 272
TRichView.AddFmt 273
TRichView.AddHotPicture 274
TRichView.AddHotPictureTag 274
TRichView.AddHotspot 275
TRichView.AddHotspotEx 275
TRichView.AddHotspotExTag 275
TRichView.AddItem 276
TRichView.AddNamedCheckpoint 271
TRichView.AddNamedCheckpointEx 271
TRichView.AddNamedCheckpointExTag 271
TRichView.AddNL 277
TRichView.AddNLATag 277
TRichView.AddNLTag 277
TRichView.AddNLWTag 277
TRichView.AddPicture 278
TRichView.AddPictureEx 278
TRichView.AddPictureExTag 278
TRichView.AddTab 280
TRichView.AddTag 277
TRichView.AddTextNL 281
TRichView.AddTextNLA 281
TRichView.AddTextNLW 281
TRichView.AllowSelection 228
TRichView.AnimationMode 228
TRichView.AppendFrom 282
TRichView.AppendRVFFromStream 283
TRichView.BackgroundBitmap 229
TRichView.BackgroundPicture 229
TRichView.BackgroundProperties 229
TRichView.BackgroundStyle 229
TRichView.BeginOleDrag 284
TRichView.BiDiMode 230
TRichView.BottomMargin 231
TRichView.Clear 286
TRichView.ClearLeft 232
TRichView.ClearLiveSpellingResults 286
TRichView.ClearRight 232
TRichView.ClearSoftPageBreaks 286
TRichView.ClientToDocument 286
TRichView.Color 233
TRichView.ConvertDocToDifferentUnits 287
TRichView.ConvertDocToEMU 287
TRichView.ConvertDocToPixels 287
TRichView.ConvertDocToTwips 287
TRichView.Copy 288
TRichView.CopyDef 288
TRichView.CopyImage 289
TRichView.CopyMarkdown 289
TRichView.CopyRTF 290
TRichView.CopyRVF 290
TRichView.CopyText 290
TRichView.CopyTextA 290
TRichView.CopyTextW 290
TRichView.CPEventKind 233
TRichView.Create 291
TRichView.Cursor 234
TRichView.DarkMode 234
TRichView.DeleteItems 291
TRichView.DeleteMarkedStyles 292
TRichView.DeleteParas 293
TRichView.DeleteSection 293
TRichView.DeleteUnusedStyles 293
TRichView.Delimiters 235
TRichView.Deselect 294
TRichView.Destroy 294
TRichView.DocObjects 235
TRichView.DocParameters 237
TRichView.DocProperties 238
TRichView.Document 238
TRichView.DocumentHeight 239
TRichView.DocumentPixelsPerInch 239
TRichView.DocumentToClient 286
TRichView.DoInPaletteMode 241
TRichView.FindCheckpointByName 294
TRichView.FindCheckpointByTag 295
TRichView.FindControlItemNo 295
TRichView.FirstItemVisible 243
TRichView.FirstJumpNo 243
TRichView.Format 295
TRichView.FormatTail 296
TRichView.GetBreakInfo 296
TRichView.GetBulletInfo 297
TRichView.GetCheckpointByNo 298
TRichView.GetCheckpointInfo 299
TRichView.GetCheckpointItemNo 299
TRichView.GetCheckpointNo 299
TRichView.GetCheckpointXY 300
TRichView.GetCheckpointY 300
TRichView.GetCheckpointYEx 301
TRichView.GetControlInfo 301
TRichView.GetFirstCheckpoint 302

TRichView.GetFocusedItem 302
TRichView.GetHotspotInfo 303
TRichView.GetItem 304
TRichView.GetItemCheckpoint 305
TRichView.GetItemClientCoords 306
TRichView.GetItemCoords 306
TRichView.GetItemCoordsEx 307
TRichView.GetItemExtraIntProperty 307
TRichView.GetItemExtraIntPropertyEx 307
TRichView.GetItemNo 309
TRichView.GetItemPara 309
TRichView.GetItemStyle 310
TRichView.GetItemTag 310
TRichView.GetItemText 310
TRichView.GetItemTextA 310
TRichView.GetItemTextW 310
TRichView.GetItemVAlign 311
TRichView.GetJumpPointItemNo 312
TRichView.GetJumpPointLocation 312
TRichView.GetJumpPointY 313
TRichView.GetLastCheckpoint 313
TRichView.GetLineNo 314
TRichView.GetListMarkerInfo 315
TRichView.GetNextCheckpoint 315
TRichView.GetOffsAfterItem 316
TRichView.GetOffsBeforeItem 317
TRichView.GetPictureInfo 318
TRichView.GetPrevCheckpoint 319
TRichView.GetRealDocumentPixelsPerInch 319
TRichView.GetSelectedImage 319
TRichView.GetSelectionBounds 320
TRichView.GetSelText 321
TRichView.GetSelTextA 321
TRichView.GetSelTextW 321
TRichView.GetTextInfo 321
TRichView.GetWordAt 322
TRichView.GetWordAtA 322
TRichView.GetWordAtW 322
TRichView.HTMLReadProperties 243
TRichView.HTMLSaveProperties 244
TRichView.InsertRVFFromStream 323
TRichView.IsFromNewLine 325
TRichView.IsParaStart 325
TRichView.ItemCount 244
TRichView.LastItemVisible 244
TRichView.LeftMargin 245
TRichView.LineCount 245
TRichView.LiveSpellingValidateWord 326
TRichView.LoadDocX 326
TRichView.LoadDocXFromStream 327
TRichView.LoadFromFile 328
TRichView.LoadFromFileEx 328
TRichView.LoadFromStream 329
TRichView.LoadFromStreamEx 329
TRichView.LoadHTML 330
TRichView.LoadHTMLFromStream 332
TRichView.LoadMarkdown 332
TRichView.LoadMarkdownFromStream 333
TRichView.LoadRTF 334
TRichView.LoadRTFFromStream 335
TRichView.LoadRVF 336
TRichView.LoadRVFFromStream 337
TRichView.LoadText 338
TRichView.LoadTextFromStream 339
TRichView.LoadTextFromStreamW 339
TRichView.LoadTextW 338
TRichView.MarkdownProperties 245
TRichView.MarkStylesInUse 339
TRichView.MaxLength 246
TRichView.MaxTextWidth 246
TRichView.MinTextWidth 246
TRichView.NoteText 247
TRichView.OnAddStyle 377
TRichView.OnAfterDrawImage 378
TRichView.OnAssignImageFileName 379
TRichView.OnCheckpointVisible 379
TRichView.OnCMHintShow 380
TRichView.OnControlAction 380
TRichView.OnCopy 381
TRichView.OnGetItemCursor 382
TRichView.OnGetSpellingSuggestions 382
TRichView.OnHTMLSaveImage 383
TRichView.OnImportFile 385
TRichView.OnImportPicture 386
TRichView.OnItemAction 387
TRichView.OnItemHint 388
TRichView.OnJump 390
TRichView.OnLoadCustomFormat 390
TRichView.OnLoadDocument 391
TRichView.OnNewDocument 391
TRichView.OnPaint 392
TRichView.OnProgress 392
TRichView.OnRVDblClick 399
TRichView.OnRVFControlNeeded 400
TRichView.OnRVFImageListNeeded 400
TRichView.OnRVFPictureNeeded 401
TRichView.OnRVMouseDown 402
TRichView.OnRVMouseMove 403

TRichView.OnRVMouseUp 403
TRichView.OnRVRightClick 404
TRichView.OnSaveComponentToFile 405
TRichView.OnSaveDocXExtra 407
TRichView.OnSaveHTMLExtra 409
TRichView.OnSaveImage2 410
TRichView.OnSaveItemToFile 411
TRichView.OnSaveParaToHTML 413
TRichView.OnSaveRTFExtra 414
TRichView.OnSelect 416
TRichView.OnSpellingCheck 416
TRichView.OnStyleTemplatesChange 417
TRichView.OnTextFound 418
TRichView.OnWriteHyperlink 418
TRichView.OnWriteObjectProperties 420
TRichView.Options 247
TRichView.PageBreaksBeforeItems 250
TRichView.Reformat 339
TRichView.RefreshListMarkers 340
TRichView.RefreshSequences 340
TRichView.RemoveCheckpoint 340
TRichView.ResetAnimation 341
TRichView.RightMargin 251
TRichView.RTFOptions 252
TRichView.RTFReadProperties 253
TRichView.RVData 254
TRichView.RVFOptions 254
TRichView.RVFParaStylesReadMode 258
TRichView.RVFTextStylesReadMode 258
TRichView.RVFWarnings 258
TRichView.SaveDocX 341
TRichView.SaveDocXToStream 342
TRichView.SaveHTML 342, 343
TRichView.SaveHTMLEx 345
TRichView.SaveHTMLToStream 346, 347
TRichView.SaveHTMLToStreamEx 348
TRichView.SaveMarkdown 348
TRichView.SaveMarkdownToStream 349
TRichView.SavePicture 349
TRichView.SaveRTF 350
TRichView.SaveRTFToStream 351
TRichView.SaveRVF 351
TRichView.SaveRVFToStream 352
TRichView.SaveText 352
TRichView.SaveTextToStream 353
TRichView.SaveTextToStreamW 353
TRichView.SaveTextW 352
TRichView.SearchText 354
TRichView.SearchTextA 354
TRichView.SearchTextW 354
TRichView.SelectAll 355
TRichView.SelectControl 355
TRichView.SelectionExists 356
TRichView.SelectionHandlesVisible 260
TRichView.SelectWordAt 356
TRichView.SetAddParagraphMode 357
TRichView.SetBreakInfo 358
TRichView.SetBulletInfo 359
TRichView.SetCheckpointInfo 360
TRichView.SetControlInfo 361
TRichView.SetFooter 362
TRichView.SetHeader 363
TRichView.SetHotspotInfo 364
TRichView.SetItemExtraIntProperty 366
TRichView.SetItemExtraIntPropertyEx 366
TRichView.SetItemExtraStrProperty 367
TRichView.SetItemExtraStrPropertyEx 367
TRichView.SetItemTag 367
TRichView.SetItemText 368
TRichView.SetItemTextA 368
TRichView.SetItemTextW 368
TRichView.SetItemVAlign 369
TRichView.SetListMarkerInfo 369
TRichView.SetPictureInfo 370
TRichView.SetSelectionBounds 372
TRichView.SingleClick 260
TRichView.StartAnimation 374
TRichView.StartLiveSpelling 374
TRichView.StopAnimation 375
TRichView.StoreSearchResult 375
TRichView.Style 260
TRichView.StyleTemplateInsertMode 260
TRichView.TabNavigation 262
TRichView.TopMargin 262
TRichView.UpdatePaletteInfo 375
TRichView.UseFMXThemes 263
TRichView.UseStyleTemplates 263
TRichView.UseVCLThemes 264
TRichView.VAlign 264
TRichView.VSmallStep 264
TRichView.WordWrap 265
TRichView.ZoomPercent 265
TRichViewEdit 469, 541
  Inserting items at position of caret 119
  InsertTextFromFile 541
  InsertTextFromFileW 541
  Moving caret to the beginning of paragraph 602
  OnDropFiles example 603

TRichViewEdit Undo 120
TRichViewEdit.AcceptDragDropFormats 478
TRichViewEdit.AcceptPasteFormats 478
TRichViewEdit.AddSpellingMenuItems 497
TRichViewEdit.AdjustControlPlacement 497
TRichViewEdit.AdjustControlPlacement2 497
TRichViewEdit.ApplyListStyle 498
TRichViewEdit.ApplyParaStyle 498
TRichViewEdit.ApplyParaStyleConversion 499
TRichViewEdit.ApplyParaStyleTemplate 500
TRichViewEdit.ApplyStyleConversion 501
TRichViewEdit.ApplyStyleTemplate 501
TRichViewEdit.ApplyTextStyle 502
TRichViewEdit.ApplyTextStyleTemplate 502
TRichViewEdit.BeginItemModify 503
TRichViewEdit.BeginUndoCustomGroup 504
TRichViewEdit.BeginUndoCustomGroup2 504
TRichViewEdit.BeginUndoGroup 505
TRichViewEdit.BeginUndoGroup2 505
TRichViewEdit.BeginUpdate 506
TRichViewEdit.CanChange 506
TRichViewEdit.CanPaste 507
TRichViewEdit.CanPasteHTML 507
TRichViewEdit.CanPasteRTF 507
TRichViewEdit.CanPasteRVF 507
TRichViewEdit.Change 508
TRichViewEdit.ChangeListLevels 508
TRichViewEdit.ChangeStyleTemplates 508
TRichViewEdit.CheckSpelling 480
TRichViewEdit.Clear 509
TRichViewEdit.ClearTextFlow 509
TRichViewEdit.ClearUndo 509
TRichViewEdit.ConvertToHotPicture 510
TRichViewEdit.ConvertToPicture 510
TRichViewEdit.Create 510
TRichViewEdit.CurItemNo 480
TRichViewEdit.CurItemStyle 481
TRichViewEdit.CurParaStyleNo 481
TRichViewEdit.CurTextStyleNo 482
TRichViewEdit.CustomCaretInterval 483
TRichViewEdit.CutDef 511
TRichViewEdit.DefaultPictureVAlign 483
TRichViewEdit.DeleteSelection 511
TRichViewEdit.Destroy 511
TRichViewEdit.EditorOptions 483
TRichViewEdit.EndItemModify 511
TRichViewEdit.EndUndoCustomGroup2 504
TRichViewEdit.EndUndoGroup2 505
TRichViewEdit.EndUpdate 512
TRichViewEdit.ForceFieldHighlight 486
TRichViewEdit.GetCheckpointAtCaret 512
TRichViewEdit.GetCurrentBreakInfo 512
TRichViewEdit.GetCurrentBulletInfo 513
TRichViewEdit.GetCurrentCheckpoint 514
TRichViewEdit.GetCurrentControlInfo 515
TRichViewEdit.GetCurrentHotspotInfo 515
TRichViewEdit.GetCurrentItem 517
TRichViewEdit.GetCurrentItemEx 517
TRichViewEdit.GetCurrentItemExtraIntProperty 518
TRichViewEdit.GetCurrentItemExtraIntPropertyEx 518
TRichViewEdit.GetCurrentItemExtraStrProperty 519
TRichViewEdit.GetCurrentItemExtraStrPropertyEx 519
TRichViewEdit.GetCurrentItemText 519
TRichViewEdit.GetCurrentItemTextA 519
TRichViewEdit.GetCurrentItemTextW 519
TRichViewEdit.GetCurrentItemVAlign 520
TRichViewEdit.GetCurrentLineCol 521
TRichViewEdit.GetCurrentMisspelling 521
TRichViewEdit.GetCurrentPictureInfo 522
TRichViewEdit.GetCurrentTag 522
TRichViewEdit.GetCurrentTextInfo 523
TRichViewEdit.InsertBreak 523
TRichViewEdit.InsertBullet 524
TRichViewEdit.InsertCheckpoint 525
TRichViewEdit.InsertControl 526
TRichViewEdit.InsertDocXFromFileEd 526
TRichViewEdit.InsertDocXFromStreamEd 527
TRichViewEdit.InsertHotPicture 528
TRichViewEdit.InsertHotspot 528
TRichViewEdit.InsertHTMLFromFileEd 529
TRichViewEdit.InsertHTMLFromStreamEd 530
TRichViewEdit.InsertItem 531
TRichViewEdit.InsertMarkdownFromFileEd 532
TRichViewEdit.InsertMarkdownFromStreamEd 532
TRichViewEdit.InsertOEMTextFromFile 533
TRichViewEdit.InsertPageBreak 534
TRichViewEdit.InsertPicture 534
TRichViewEdit.InsertRTFFromFileEd 536
TRichViewEdit.InsertRTFFromStreamEd 536
TRichViewEdit.InsertRVFFromFileEd 537
TRichViewEdit.InsertRVFFromStreamEd 538
TRichViewEdit.InsertStringATag 539
TRichViewEdit.InsertStringTag 539
TRichViewEdit.InsertStringWTag 539
TRichViewEdit.InsertTab 540
TRichViewEdit.InsertText 540
TRichViewEdit.InsertTextA 540
TRichViewEdit.InsertTextW 540

TRichViewEdit.KeyboardType 487
TRichViewEdit.LiveSpellingMode 487
TRichViewEdit.Modified 487
TRichViewEdit.MoveCaret 542
TRichViewEdit.OffsetInCurItem 488
TRichViewEdit.OnAfterOleDrop 580
TRichViewEdit.OnBeforeOleDrop 580
TRichViewEdit.OnCaretGetOut 581
TRichViewEdit.OnCaretMove 582
TRichViewEdit.OnChange 582
TRichViewEdit.OnChanging 582
TRichViewEdit.OnCheckStickingItems 583
TRichViewEdit.OnCurParaStyleChanged 583
TRichViewEdit.OnCurTextStyleChanged 584
TRichViewEdit.OnDrawCurrentLine 584
TRichViewEdit.OnDrawCustomCaret 585
TRichViewEdit.OnDropFile 586
TRichViewEdit.OnDropFiles 586
TRichViewEdit.OnItemResize 587
TRichViewEdit.OnItemTextEdit 588
TRichViewEdit.OnMeasureCustomCaret 588
TRichViewEdit.OnOleDragEnter 589
TRichViewEdit.OnOleDragLeave 589
TRichViewEdit.OnOleDragOver 590
TRichViewEdit.OnOleDrop 591
TRichViewEdit.OnParaStyleConversion 592
TRichViewEdit.OnPaste 594
TRichViewEdit.OnSaveCustomFormat 595
TRichViewEdit.OnSmartPopupClick 595
TRichViewEdit.OnStyleConversion 595
TRichViewEdit.Paste 543
TRichViewEdit.PasteBitmap 544
TRichViewEdit.PasteGraphicFile 544
TRichViewEdit.PasteHTML 545
TRichViewEdit.PasteMarkdown 546
TRichViewEdit.PasteMetafile 546
TRichViewEdit.PasteRTF 547
TRichViewEdit.PasteRVF 547
TRichViewEdit.PasteText 548
TRichViewEdit.PasteTextA 548
TRichViewEdit.PasteTextW 548
TRichViewEdit.PasteURL 549
TRichViewEdit.ReadOnly 488
TRichViewEdit.Redo 549
TRichViewEdit.RedoAction 549
TRichViewEdit.RedoName 550
TRichViewEdit.RemoveCheckpointAtCaret 550
TRichViewEdit.RemoveCheckpointEd 550
TRichViewEdit.RemoveCurrentCheckpoint 551
TRichViewEdit.RemoveCurrentPageBreak 551
TRichViewEdit.RemoveLists 552
TRichViewEdit.ResizeControl 552
TRichViewEdit.ResizeCurrentControl 552
TRichViewEdit.SearchText 553
TRichViewEdit.SearchTextA 553
TRichViewEdit.SearchTextW 553
TRichViewEdit.SelectCurrentLine 554
TRichViewEdit.SelectCurrentWord 555
TRichViewEdit.SetBackgroundImageEd 556
TRichViewEdit.SetBackgroundPropertyEd 557
TRichViewEdit.SetBreakInfoEd 557
TRichViewEdit.SetBulletInfoEd 558
TRichViewEdit.SetCheckpointInfoEd 560
TRichViewEdit.SetControlInfoEd 560
TRichViewEdit.SetCurrentBreakInfo 561
TRichViewEdit.SetCurrentBulletInfo 562
TRichViewEdit.SetCurrentCheckpointInfo 563
TRichViewEdit.SetCurrentControlInfo 564
TRichViewEdit.SetCurrentHotspotInfo 565
TRichViewEdit.SetCurrentItemExtraIntProperty 566
TRichViewEdit.SetCurrentItemExtraIntPropertyEx 566
TRichViewEdit.SetCurrentItemExtraStrProperty 567
TRichViewEdit.SetCurrentItemExtraStrPropertyEx 567
TRichViewEdit.SetCurrentItemText 568
TRichViewEdit.SetCurrentItemTextA 568
TRichViewEdit.SetCurrentItemTextW 568
TRichViewEdit.SetCurrentItemVAlign 569
TRichViewEdit.SetCurrentPictureInfo 569
TRichViewEdit.SetCurrentTag 570
TRichViewEdit.SetFloatPropertyEd 555
TRichViewEdit.SetHotspotInfoEd 570
TRichViewEdit.SetIntPropertyEd 555
TRichViewEdit.SetItemExtraIntPropertyEd 572
TRichViewEdit.SetItemExtraIntPropertyExEd 572
TRichViewEdit.SetItemExtraStrPropertyEd 573
TRichViewEdit.SetItemExtraStrPropertyExEd 573
TRichViewEdit.SetItemTagEd 574
TRichViewEdit.SetItemTextEd 574
TRichViewEdit.SetItemTextEdA 574
TRichViewEdit.SetItemTextEdW 574
TRichViewEdit.SetItemVAlignEd 575
TRichViewEdit.SetPictureInfoEd 576
TRichViewEdit.SetStrPropertyEd 555
TRichViewEdit.SetUndoGroupMode 577
TRichViewEdit.SmartPopupProperties 489
TRichViewEdit.SmartPopupVisible 489
TRichViewEdit.TopLevelEditor 489
TRichViewEdit.Undo 577

TRichViewEdit.UndoAction 577
TRichViewEdit.UndoLimit 490
TRichViewEdit.UndoName 578
TRichViewRVData 1014
TRichViewUnicodeInput 1118
TRV_AfterImportGraphicsProc 1136
TRV_CreateGraphicsFunction 1136
TRVAddStyleEvent 377
TRVAfterApplyStyleEvent 774
TRVAfterDrawImageEvent 378
TRVAfterPrintImageEvent 885
TRVAlignment 829
TRVAnimationMode 228
TRVAnsiString 1056
TRVApplyStyleColorEvent 775
TRVApplyStyleEvent 774
TRVAssignImageFileNameEvent 379
TRVBackgroundAttachment 1022
TRVBackgroundHorizontalAlignment 1023
TRVBackgroundProperties 1021
TRVBackgroundProperties.Attachment 1022
TRVBackgroundProperties.HorizontalAlignment 1023
TRVBackgroundProperties.HorizontalOffset 1023
TRVBackgroundProperties.HorizontalOffsetKind 1023
TRVBackgroundProperties.HorizontalRepeat 1024
TRVBackgroundProperties.HorizontalSizeKind 1024
TRVBackgroundProperties.ImageHeight 1025
TRVBackgroundProperties.ImageWidth 1024
TRVBackgroundProperties.VerticalAlignment 1024
TRVBackgroundProperties.VerticalOffset 1025
TRVBackgroundProperties.VerticalOffsetKind 1025
TRVBackgroundProperties.VerticalRepeat 1025
TRVBackgroundProperties.VerticalSizeKind 1025
TRVBackgroundProperties.Visible 1025
TRVBackgroundPropertyId 1056
TRVBackgroundRect 851
TRVBackgroundRect.BorderOffsets 851
TRVBackgroundRect.Color 852
TRVBackgroundRect.Opacity 852
TRVBackgroundRepeat 1057
TRVBackgroundSize 1058
TRVBackgroundVerticalAlignment 1024
TRVBiDiMode 1058
TRVBooleanRect 1018
TRVBorder 852
TRVBorder.BorderOffsets 853
TRVBorder.Color 853
TRVBorder.InternalWidth 853
TRVBorder.Style 854
TRVBorder.VisibleBorders 854
TRVBorder.Width 854
TRVBorderStyle 1059
TRVBoxBackgroundRect 995
TRVBoxBorder 995
TRVBoxHeightType 995
TRVBoxPosition 997
TRVBoxPosition.HorizontalAlignment 999
TRVBoxPosition.HorizontalAnchor 998
TRVBoxPosition.HorizontalOffset 999
TRVBoxPosition.HorizontalPositionKind 999
TRVBoxPosition.PositionInText 1001
TRVBoxPosition.RelativeToCell 1002
TRVBoxPosition.VerticalAlignment 1003
TRVBoxPosition.VerticalAnchor 1002
TRVBoxPosition.VerticalOffset 1003
TRVBoxPosition.VerticalPositionKind 1003
TRVBoxProperties 994
TRVBoxProperties.Background 995
TRVBoxProperties.Border 995
TRVBoxProperties.Height 995
TRVBoxProperties.HeightType 995
TRVBoxProperties.VAlign 996
TRVBoxProperties.Width 996
TRVBoxProperties.WidthType 996
TRVBoxWidthType 996
TRVBreakItemInfo 899
TRVBreakStyle 1059
TRVBulletItemInfo 899
TRVCalloutType 658
TRVCanvas 1018
TRVCaretMovement 542
TRVCaretWidthMode 1108
TRVCellEditingEvent 948
TRVCellVAlign 1061
TRVChat 640
TRVChat.AddMessage 645
TRVChat.AddStyles 646
TRVChat.Callout 641
TRVChat.ChangeMessageImage 646
TRVChat.ChangeMessageText 647
TRVChat.ChatMaxWidth 641
TRVChat.ChatRichView 642
TRVChat.ClearMessages 647
TRVChat.CopyToClipboard 647
TRVChat.DateLine 642
TRVChat.DeleteMessage 647
TRVChat.FileIcons 642
TRVChat.FindMessages 648

TRVChat.Fonts 642
TRVChat.GetAutoQuote 648
TRVChat.GetAutoQuoteFromItemNo 648
TRVChat.GetFirstVisibleMessage 649
TRVChat.GetImageAtCursor 649
TRVChat.GetMessage 649
TRVChat.GetMessageAndImage 649
TRVChat.GetMessageAtClientCoords 649
TRVChat.GetMessageAtCursor 649
TRVChat.GetMessageAtDocumentCoords 649
TRVChat.GetMessageAtScreenCoords 649
TRVChat.GetMessageImage 650
TRVChat.GetMessageImageEx 650
TRVChat.GetMessageItemNo 650
TRVChat.GetMessagesAsText 651
TRVChat.GetMessageTextItemNo 650
TRVChat.GetRangeOfSelectedMessages 651
TRVChat.GetSelectedMessage 651
TRVChat.GetSelectedTextForQuote 652
TRVChat.HasSelectedTextForQuote 652
TRVChat.HighlightedMessage 642
TRVChat.InsertMessage 652
TRVChat.LinkColor 643
TRVChat.MarkMessageAsDeleted 653
TRVChat.Messages 643
TRVChat.MessageStatusLine 643
TRVChat.MoveMessage 653
TRVChat.OnClickDate 654
TRVChat.OnClickFile 654
TRVChat.OnClickImage 655
TRVChat.OnClickLink 655
TRVChat.OnClickUser 655
TRVChat.OnHighlightedMessageChange 655
TRVChat.Quote 643
TRVChat.ScrollToMessage 654
TRVChat.Texts 643
TRVChat.UpdateParaStyles 654
TRVChat.UpdateTextStyles 654
TRVChat.UserPicture 644
TRVChat.Users 644
TRVChatCalloutProperties 641, 657
TRVChatCalloutProperties.ArrowHeight 657
TRVChatCalloutProperties.ArrowWidth 657
TRVChatCalloutProperties.CalloutType 658
TRVChatCalloutProperties.CornerRadiusLarge 658
TRVChatCalloutProperties.CornerRadiusSmall 658
TRVChatCalloutProperties.Margin 658
TRVChatCalloutProperties.MaxWidth 658
TRVChatCalloutProperties.MinOppositeMargin 659
TRVChatCalloutProperties.MinWidth 658
TRVChatCalloutProperties.Padding 659
TRVChatCalloutProperties.ShadowOpacity 659
TRVChatCalloutProperties.Spacing 659
TRVChatDateProperties 660
TRVChatDateProperties.BackColor 660
TRVChatDateProperties.DateFormatNew 660
TRVChatDateProperties.DateFormatOld 660
TRVChatDateProperties.MarginAbove 661
TRVChatDateProperties.MarginBelow 661
TRVChatDateProperties.PaddingHorizontal 661
TRVChatDateProperties.PaddingVertical 661
TRVChatFileCollection 661
TRVChatFileCollection.Items 662
TRVChatFileIconsProperties 662
TRVChatFileIconsProperties.IconColor 662
TRVChatFileIconsProperties.IconPadding 662
TRVChatFileIconsProperties.IconSize 662
TRVChatFileIconsProperties.PictureArchive 663
TRVChatFileIconsProperties.PictureAudio 663
TRVChatFileIconsProperties.PictureDocument 663
TRVChatFileIconsProperties.PictureDownload 663
TRVChatFileIconsProperties.PictureImage 663
TRVChatFileIconsProperties.PictureVideo 663
TRVChatFileIconsProperties.ScalePictures 663
TRVChatFileItem 663
TRVChatFileItem.DisplayName 664
TRVChatFileItem.FileName 664
TRVChatFileItem.FileSize 664
TRVChatFileItem.FileType 664
TRVChatFileType 1061
TRVChatFonts 664
TRVChatFonts.DateFont 664
TRVChatFonts.DeletedMessageFont 664
TRVChatFonts.FileInfoFont 664
TRVChatFonts.FileNameFont 664
TRVChatFonts.MessageFont 664
TRVChatFonts.QuoteFont 664
TRVChatFonts.SetDefaultFontName 664
TRVChatFonts.SetFontName 664
TRVChatFonts.TimeFont 664
TRVChatFonts.UserNameFont 664
TRVChatItemId 1061
TRVChatMessage 665
TRVChatMessage.DateTime 666
TRVChatMessage.EditCount 666
TRVChatMessage.Files 668
TRVChatMessage.GlobalIndex 667
TRVChatMessage.Id 666

TRVChatMessage.Quote 667
TRVChatMessage.State 667
TRVChatMessage.Text 667
TRVChatMessage.User 667
TRVChatMessageCollection 668
TRVChatMessageCollection.FindByDate 669
TRVChatMessageCollection.FindByGlobalIndex 669
TRVChatMessageCollection.FindById 669
TRVChatMessageCollection.FindInsertionPlace 670
TRVChatMessageCollection.FindItemByDate 669
TRVChatMessageCollection.FindItemByGlobalIndex 669
TRVChatMessageCollection.FindItemById 669
TRVChatMessageCollection.Items 668
TRVChatMessageExtra 670
TRVChatMessageExtra.AutoDestroy 670
TRVChatMessageExtraFiles 670
TRVChatMessageExtraFiles.AddFile 671
TRVChatMessageExtraFiles.AssignDefaultFileTypes 672
TRVChatMessageExtraFiles.Create 672
TRVChatMessageExtraFiles.Files 671
TRVChatMessageExtraImage 672
TRVChatMessageExtraImage.Create 673
TRVChatMessageExtraImage.Image 673
TRVChatMessageExtraImage.ImageText 673
TRVChatMessageExtraImage.MustAddCopyOfImage 673
TRVChatMessageExtraQuote 673
TRVChatMessageExtraQuote.Create 674
TRVChatMessageExtraQuote.Msg 674
TRVChatMessageExtraQuote.Text 674
TRVChatMessageExtraQuote.User 674
TRVChatMessageInsertionInfo 675
TRVChatMessageQuote 675
TRVChatMessageQuote.Msg 676
TRVChatMessageQuote.QuoteType 676
TRVChatMessageQuote.Text 676
TRVChatMessageQuote.User 676
TRVChatMessageState 1062
TRVChatMessageStatusLine 676
TRVChatMessageStatusLine.FailedImageIndex 679
TRVChatMessageStatusLine.Images 677
TRVChatMessageStatusLine.ImagesHeight 677
TRVChatMessageStatusLine.ImagesWidth 677
TRVChatMessageStatusLine.MinStatusLineHeight 677
TRVChatMessageStatusLine.NewImageIndex 679
TRVChatMessageStatusLine.ReadImageIndex 679
TRVChatMessageStatusLine.SentImageIndex 679
TRVChatMessageStatusLine.StatusLineContent 678
TRVChatMessageStatusLine.StatusLineLocation 678
TRVChatQuoteProperties 679
TRVChatQuoteProperties.CompactQuotes 679
TRVChatQuoteProperties.CornerRadius 680
TRVChatQuoteProperties.MaxAutoQuoteLength 680
TRVChatQuoteProperties.Padding 680
TRVChatQuoteType 1062
TRVChatStatusLineItem 678
TRVChatStatusLineItems 678
TRVChatUITexts 680
TRVChatUITexts.B 680
TRVChatUITexts.DeletedMessage 680
TRVChatUITexts.Edited 680
TRVChatUITexts.GB 680
TRVChatUITexts.KB 680
TRVChatUITexts.MB 680
TRVChatUser 681
TRVChatUser.FillColor 682
TRVChatUser.Id 682
TRVChatUser.IsRightSide 682
TRVChatUser.MessageInfoColor 682
TRVChatUser.Name 682
TRVChatUser.NameColor 682
TRVChatUser.Picture 683
TRVChatUser.QuoteFillColor 683
TRVChatUser.ShowNameInMessages 683
TRVChatUser.ShowPicture 683
TRVChatUserCollection 683
TRVChatUserCollection.FindById 684
TRVChatUserCollection.FindItemById 684
TRVChatUserCollection.Items 684
TRVChatUserPicturePosition 685
TRVChatUserPictureProperties 684
TRVChatUserPictureProperties.Position 685
TRVChatUserPictureProperties.Size 685
TRVCheckPointVisibleEvent 379
TRVClickDateEvent 654
TRVClickFileEvent 654
TRVClickImageEvent 655
TRVClickLinkEvent 655
TRVClickMode 891
TRVClickUserEvent 655
TRVCMHintShowEvent 380
TRVCodePage 1054, 1062
TRVColor 1063
TRVColorMode 878, 1063
TRVControlAction 380
TRVControlActionEvent 380
TRVControlData 1014
TRVControlItemInfo 899
TRVControlsPainter 1019

TRVCoord 1064
TRVCoordPoint 1064
TRVCoordRect 1064
TRVCoordSize 1064
TRVCSSSavingType 438
TRVCustomFormatEvent 390, 595
TRVCustomMathEquationManager 1020
SaveMathMLInHTML 1021
SaveOMMLInDocX 1021
TRVCustomMathEquationManager.SaveMathMLInHTML 1021
TRVCustomMathEquationManager.SaveOMMLInDocX 1021
TRVDataSourceLink 685
TRVDataSourceLink.DataSource 686
TRVDataSourceLink.Editor 686
TRVDataSourceLink.Execute 687
TRVDBFieldFormat 1064
TRVDblClickEvent 399
TRVDefaultLoadProperties 1026
TRVDefaultLoadProperties.DefaultBulletFontName 1026
TRVDefaultLoadProperties.DefaultFontName 1026
TRVDefaultLoadProperties.DefaultFontSizeDouble 1027
TRVDefaultLoadProperties.DefaultFontSizesDouble 1027
TRVDefaultLoadProperties.DefaultMonoSpacedFontName 1028
TRVDefaultLoadProperties.DefaultProperties 1028
TRVDefaultLoadProperties.DefaultUnicodeSymbolFontName 1028
TRVDefaultLoadProperties.GenericFontNames 1028
TRVDefaultLoadProperties.ListMarkerIndentPix 1031
TRVDeleteUnusedStylesData 1031
TRVDisplayOption 1065
TRVDisplayOptions 1065
TRVDocObject 235
TRVDocObjectCollection 235
TRVDocParameters 237, 422
TRVDocParameters.Author 424
TRVDocParameters.BottomMargin 424
TRVDocParameters.Comments 424
TRVDocParameters.ConvertToUnit 428
TRVDocParameters.FacingPages 424
TRVDocParameters.FooterY 425
TRVDocParameters.HeaderY 425
TRVDocParameters.LeftMargin 425
TRVDocParameters.MirrorMargins 426
TRVDocParameters.Orientation 426
TRVDocParameters.PageHeight 426
TRVDocParameters.PageWidth 426
TRVDocParameters.Reset 429
TRVDocParameters.ResetLayout 429
TRVDocParameters.RightMargin 426
TRVDocParameters.Title 427
TRVDocParameters.TitlePage 427
TRVDocParameters.TopMargin 427
TRVDocParameters.Units 428
TRVDocParameters.UnitsPerInch 429
TRVDocParameters.ZoomMode 428
TRVDocParameters.ZoomPercent 428
TRVDocRotation 1065
TRVDocumentProperty 429
TRVDocumentProperty.AllowMarkdown 430
TRVDocumentProperty.AutoDeleteUnusedStyles 430
TRVDocumentProperty.CodePage 431
TRVDocumentProperty.FieldFormat 431
TRVDocXErrorCode 462
TRVDocXSaveArea 407
TRVDragDropFormat 478
TRVDragDropFormats 478
TRVDrawCheckpointEvent 776, 885
TRVDrawCurrentLineEvent 584
TRVDrawCustomCaretEvent 585
TRVDrawHyperlinkEvent 886
TRVDrawPageBreakEvent 778
TRVDrawParaRectEvent 780
TRVDrawStyleTextEvent 781
TRVDrawTextBackEvent 783
TRVDropFileAction 586
TRVDropFileEvent 586
TRVDropFilesEvent 586
TRVDualLength 1066
TRVDualSignedLength 1066
TRVDualSignedLengthKind 1066
TRVEditRVData 1014
TRVEndnoteItemInfo 899, 985
TRVEndnoteItemInfo.Create 987
TRVEndnoteItemInfo.CreateEx 987
TRVEnumScope 1066
TRVESearchOption 1066
TRVESearchOptions 1066
TRVExtraItemProperty 1067
TRVExtraItemStrProperty 1072
TRVFControlNeededEvent 400
TRVFieldContext 393, 398
TRVFieldResultData 393, 398
TRVFImageListNeededEvent 400
TRVFixMarginsMode 703
TRVFloatHorizontalAlignment 999

TRVFloatPosition 1073
TRVFloatPositionInText 1001
TRVFloatPositionKind 1073
TRVFloatProperty 1074
TRVFloatSize 1074
TRVFloatVerticalAlignment 1003
TRVFMXCanvas 1018
TRVFont 1032
TRVFontCharset 1075
TRVFontInfoClass 770
TRVFontInfoProperties 1075
TRVFontInfoProperty 1075
TRVFontSize 1076
TRVFontSizePrecision 434
TRVFontStyle 818
TRVFontStyles 818
TRVFootnoteItemInfo 899, 988, 990
TRVFootnoteItemInfo.Create 990
TRVFootnoteItemInfo.CreateEx 990
TRVFootOrEndnoteItemInfo 982
TRVFOption 254
TRVFOptions 254
TRVFormattingDefaultItem 1028
TRVFormattingType 1028
TRVFPictureNeededEvent 401
TRVFReaderStyleMode 1076
TRVFullLineItemInfo 899
TRVFWarning 258
TRVFWarnings 258
TRVGetItemCursorEvent 382
TRVGetOutDirection 581
TRVGetPageNumberEvent 728
TRVGetSpellingSuggestionsEvent 382
TRVGetSystemColorFunction 1125
TRVGraphic 1032
TRVGraphicClass 1033
TRVGraphicFM 1032
TRVGraphicHandler 1033
TRVGraphicItemInfo 899
TRVGraphicType 1077
TRVHFType 1078
TRVHorizontalAnchor 998
TRVHorizontalCaretPosition 1108
TRVHotGraphicItemInfo 899
TRVHotspotItemInfo 899
TRVHoverEffect 812
TRVHoverEffects 812
TRVHTMLCSSSaveOption 438
TRVHTMLCSSSaveOptions 438
TRVHTMLEncoding 1078
TRVHTMLLength 1079
TRVHTMLListMarkerSavingType 444
TRVHTMLReaderProperties 432
TRVHTMLReaderProperties.BasePathLinks 433
TRVHTMLReaderProperties.EnforceListLevels 433
TRVHTMLReaderProperties.FontSizePrecision 434
TRVHTMLReaderProperties.IDAsCheckpoints 434
TRVHTMLReaderProperties.ReadBackground 434
TRVHTMLReaderProperties.ReadDocParameters 435
TRVHTMLReaderProperties.SkipHiddenText 435
TRVHTMLReaderProperties.XHTMLCheck 436
TRVHTMLSaveArea 409
TRVHTMLSaveImageEvent 383
TRVHTMLSaveImageOption 442
TRVHTMLSaveImageOptions 442
TRVHTMLSaveProperties 436
TRVHTMLSaveProperties.CheckpointsPrefix 437
TRVHTMLSaveProperties.CSSOptions 438
TRVHTMLSaveProperties.CSSSavingType 438
TRVHTMLSaveProperties.Default0Style 439
TRVHTMLSaveProperties.Encoding 439
TRVHTMLSaveProperties.EncodingStr 439
TRVHTMLSaveProperties.ExternalCSSFileName 440
TRVHTMLSaveProperties.ExtraStyles 440
TRVHTMLSaveProperties.HTMLSavingType 441
TRVHTMLSaveProperties.HTMLVersion 441
TRVHTMLSaveProperties.ImageOptions 442
TRVHTMLSaveProperties.ImagesPrefix 443
TRVHTMLSaveProperties.ListMarkerSavingType 444
TRVHTMLSaveProperties.SaveHeaderAndFooter 445
TRVHTMLSaveProperties.UseCheckpointNames 445
TRVHTMLSavingPart 1079
TRVHTMLSavingType 441
TRVHTMLVersion 441
TRVImportFileEvent 385
TRVImportPictureEvent 386
TRVIntegerList 1036
TRVIntProperty 1080
TRVItemAction 387
TRVItemActionEvent 387
TRVItemBackgroundStyle 1080
TRVItemFactory 1036
TRVItemFormattedData 1014
TRVItemHintEvent 388
TRVItemResizeEvent 587
TRVItemTextEditEvent 588
TRVLabelItemInfo 899, 970
TRVLabelItemInfo.Alignment 971

TRVLabelItemInfo.Create 973
TRVLabelItemInfo.CreateEx 973
TRVLabelItemInfo.Cursor 971
TRVLabelItemInfo.MinWidth 971
TRVLabelItemInfo.ProtectTextStyleNo 972
TRVLabelItemInfo.RemoveInternalLeading 972
TRVLabelItemInfo.RemoveSpaceBelow 972
TRVLabelItemInfo.Text 972
TRVLabelItemInfo.TextStyleNo 973
TRVLastLineAlignment 829
TRVLength 1081
TRVLineSpacingType 833
TRVLineSpacingValue 832
TRVLineWrapMode 753
TRVListInfo 841
TRVListInfo.AllNumbered 842
TRVListInfo.HasNumbering 843
TRVListInfo.Levels 842
TRVListInfo.OneLevelPreview 842
TRVListInfoClass 770
TRVListInfos 795
TRVListInfos.Add 797
TRVListInfos.AssignTo 797
TRVListInfos.FindStyleWithLevels 796
TRVListInfos.FindSuchStyle 796
TRVListInfos.Items 795
TRVListLevel 859
TRVListLevel.FirstIndent 860
TRVListLevel.Font 861
TRVListLevel.FormatString 861
TRVListLevel.FormatStringW 862
TRVListLevel.HasNumbering 868
TRVListLevel.ImageHeight 863
TRVListLevel.ImageIndex 863
TRVListLevel.ImageList 863
TRVListLevel.ImageWidth 863
TRVListLevel.LeftIndent 864
TRVListLevel.ListType 864
TRVListLevel.MarkerAlignment 866
TRVListLevel.MarkerIndent 866
TRVListLevel.Options 867
TRVListLevel.Picture 868
TRVListLevel.StartFrom 868
TRVListLevelCollection 858
TRVListLevelCollection.Add 859
TRVListLevelCollection.Items 859
TRVListLevelOption 867
TRVListLevelOptions 867
TRVListType 864
TRVLiveSpellingMode 487
TRVLoadFormat 1081
TRVLongOperation 392
TRVMarginsPen 892
TRVMarkdownDefaultItemProperties 445, 452
TRVMarkdownDefaultItemProperties.BlockQuoteBackColor 446
TRVMarkdownDefaultItemProperties.BreakColor 446
TRVMarkdownDefaultItemProperties.CodeBlockBackColor 446
TRVMarkdownDefaultItemProperties.ImageVAlign 447
TRVMarkdownDefaultItemProperties.ListLeftIndentPix 447
TRVMarkdownDefaultItemProperties.ListMarkerIndentPix 447
TRVMarkdownDefaultItemProperties.TableCellWidthPix 447
TRVMarkdownExtension 452
TRVMarkdownExtensions 452
TRVMarkdownLoadOption 453
TRVMarkdownLoadOptions 453
TRVMarkdownProperties 447
TRVMarkdownProperties.DefaultItemProperties 452
TRVMarkdownProperties.Extensions 452
TRVMarkdownProperties.ImagesPrefix 453
TRVMarkdownProperties.LineWidth 453
TRVMarkdownProperties.LoadOptions 453
TRVMarkdownProperties.NotesSavedAsFootnotes 455
TRVMarkdownProperties.SaveHiddenText 456
TRVMarkdownProperties.SaveOptions 456
TRVMarkdownSaveOption 456
TRVMarkdownSaveOptions 456
TRVMarkerAlignment 866
TRVMarkerItemInfo 899
TRVMathDocObject 1037
TRVMathItemInfo 974
TRVMathItemInfo.Create 978
TRVMathItemInfo.DisplayInline 975
TRVMathItemInfo.FontName 976
TRVMathItemInfo.FontSizeDouble 976
TRVMathItemInfo.Text 977
TRVMathItemInfo.TextColor 977
TRVMeasureCustomCaretEvent 588
TRVMouseEvent 402, 403
TRVMouseMoveEvent 403
TRVNonTextItemInfo 899
TRVNoteData 1014
TRVNoteReferenceItemInfo 899, 1006
TRVNoteReferenceItemInfo.Create 1008
TRVNoteReferenceItemInfo.CreateEx 1008

TRVNoteType 455
TRVNoteTypes 455
TRVObjectExportProperties 420
TRVOfficeCnvList 696
TRVOfficeCnvList.Count 696
TRVOfficeCnvList.Items 696
TRVOfficeConverter 687
TRVOfficeConverter.Create 692
TRVOfficeConverter.Destroy 692
TRVOfficeConverter.ErrorCode 689
TRVOfficeConverter.ExcludeDocXExportConverter 690
TRVOfficeConverter.ExcludeDocXImportConverter 690
TRVOfficeConverter.ExcludeHTMLExportConverter 690
TRVOfficeConverter.ExcludeHTMLImportConverter 690
TRVOfficeConverter.ExportConverters 690
TRVOfficeConverter.ExportRTF 692
TRVOfficeConverter.ExportRV 693
TRVOfficeConverter.ExtensionsInFilter 690
TRVOfficeConverter.GetExportFilter 693
TRVOfficeConverter.GetImportFilter 693
TRVOfficeConverter.ImportConverters 691
TRVOfficeConverter.ImportRTF 694
TRVOfficeConverter.ImportRV 694
TRVOfficeConverter.IsValidImporter 695
TRVOfficeConverter.OnConverting 695
TRVOfficeConverter.PreviewMode 691
TRVOfficeConverter.Stream 691
TRVOfficeConverterInfo 696
TRVOfficeConverterInfo.Filter 696
TRVOfficeConverterInfo.Name 696
TRVOfficeConverterInfo.Path 696
TRVOleDragEnterEvent 589
TRVOleDragOverEvent 590
TRVOleDropEffect 1081
TRVOleDropEffects 1081
TRVOleDropEvent 591
TRVOnCaretGetOutEvent 581
TRVOpacity 1082
TRVOption 247
TRVOptions 247
TRVPageBreakType 778
TRVPageCountItemInfo 1011
TRVPageCountItemInfo.Create 1012
TRVPageCountItemInfo.CreateEx 1012
TRVPageNumberItemInfo 1010
TRVPageNumberItemInfo.Create 1010
TRVPageNumberItemInfo.CreateEx 1011
TRVPageNumberType 1082
TRVPagePrepaintEvent 718, 719
TRVPageSet 1083
TRVPaintEvent 378, 392
TRVPaletteAction 241
TRVPAPen 893
TRVParagraphMarks 1126
TRVParaInfoClass 770
TRVParaInfoProperties 1083
TRVParaInfoProperty 1083
TRVParaOption 833
TRVParaOptions 833
TRVPasteEvent 594
TRVPDFMetadata 1084
TRVPenStyle 1084
TRVPicture 1037
TRVPictureFM 1037
TRVPixel96Length 1085
TRVPixelLength 1085
TRVPrint 696
TRVPrint.AssignDocParameters 708
TRVPrint.AssignSource 709
TRVPrint.BestDPI 702
TRVPrint.ClipMargins 703
TRVPrint.ContinuousPrint 709
TRVPrint.ConvertToUnits 710
TRVPrint.Create 710
TRVPrint.DrawPage 710
TRVPrint.DrawPageAt 710
TRVPrint.DrawPreview 711
TRVPrint.FacingPages 703
TRVPrint.FixMarginsMode 703
TRVPrint.FooterY 704
TRVPrint.FormatPages 711
TRVPrint.GetFooterRect 712
TRVPrint.GetHeaderRect 712
TRVPrint.HeaderY 705
TRVPrint.MakePreview 712
TRVPrint.MakeScaledPreview 713
TRVPrint.Margins 705
TRVPrint.MirrorMargins 706
TRVPrint.OnFormatting 717
TRVPrint.OnPagePostpaint 718
TRVPrint.OnPagePrepaint 719
TRVPrint.OnSendingToPrinter 719
TRVPrint.Preview100PercentHeight 706
TRVPrint.Preview100PercentWidth 706
TRVPrint.Print 713
TRVPrint.PrintPages 714
TRVPrint.SavePDF 715
TRVPrint.SetFooter 715

TRVPrint.SetHeader 716
TRVPrint.TitlePage 706
TRVPrint.Units 707
TRVPrint.VirtualPrinter 707
TRVPrintComponentEvent 887
TRVPrinterDPI 702
TRVPrintingEvent 717, 719
TRVPrintingStep 1085
TRVPrintPreview 635
TRVPrintPreview.CachePageImage 638
TRVPrintPreview.RVPrint 638
TRVProgressEvent 392
TRVProgressStage 392
TRVProtectOption 815
TRVProtectOptions 815
TRVRawByteString 1085
TRVReaderStyleMode 1086
TRVReaderUnicode 467
TRVReadFieldEvent 393, 398
TRVReadHyperlink 395
TRVRect 1038
TRVRectItemInfo 899
TRVReportHelper 721
TRVReportHelper.AllowTransparentDrawing 723
TRVReportHelper.DrawPage 725
TRVReportHelper.DrawPageAt 725
TRVReportHelper.Finished 726
TRVReportHelper.FormatNextPage 726
TRVReportHelper.GetLastPageHeight 726
TRVReportHelper.Init 727
TRVReportHelper.OnGetPageNumber 728
TRVReportHelper.RichView 724
TRVReportHelper.SavePDF 727
TRVReportHelper.TargetPixelsPerInch 723
TRVReportHelper.UpdatePageNumbers 727
TRVReportHelperWithHeaderFooters 1038
TRVReportHelperWithHeaderFooters.AssignToRVPrint 1040
TRVReportHelperWithHeaderFooters.Create 1040
TRVReportHelperWithHeaderFooters.GetFooter 1040
TRVReportHelperWithHeaderFooters.GetHeader 1040
TRVReportHelperWithHeaderFooters.MainDoc 1039
TRVReportHelperWithHeaderFooters.Reset 1041
TRVReportHelperWithHeaderFooters.UseStyleTemplates 1039
TRVRightClickEvent 404
TRVRTFErrorCode 466
TRVRTFFieldResultAction 393
TRVRTFHighlight 460
TRVRTFOption 252
TRVRTFOptions 252
TRVRTFReaderProperties 457
TRVRTFReaderProperties.AutoHideTableGridLines 459
TRVRTFReaderProperties.BasePathLinks 459
TRVRTFReaderProperties.CanReuseNumberedLists 459
TRVRTFReaderProperties.CharsetForUnicode 460
TRVRTFReaderProperties.ConvertHighlight 460
TRVRTFReaderProperties.ConvertSymbolFonts 461
TRVRTFReaderProperties.DefCharsetCodePage 461
TRVRTFReaderProperties.DocXErrorCode 462
TRVRTFReaderProperties.ExtractMetafileBitmaps 462
TRVRTFReaderProperties.IgnoreBookmarks 463
TRVRTFReaderProperties.IgnoreFields 463
TRVRTFReaderProperties.IgnoreNotes 463
TRVRTFReaderProperties.IgnorePictures 463
TRVRTFReaderProperties.IgnoreSequences 463
TRVRTFReaderProperties.IgnoreTables 464
TRVRTFReaderProperties.LineBreaksAsParagraphs 464
TRVRTFReaderProperties.ParaStyleMode 464
TRVRTFReaderProperties.ParaStyleNo 465
TRVRTFReaderProperties.ReadDocParameters 465
TRVRTFReaderProperties.ReadNoteNumberingType 465
TRVRTFReaderProperties.RTFErrorCode 466
TRVRTFReaderProperties.SkipHiddenText 466
TRVRTFReaderProperties.TextStyleMode 466
TRVRTFReaderProperties.TextStyleNo 467
TRVRTFReaderProperties.UnicodeMode 467
TRVRTFReaderProperties.UseCharsetForUnicode 468
TRVRTFReaderProperties.UseHypertextStyles 468
TRVRTFReaderProperties.UseSingleCellPadding 468
TRVRTFSaveArea 414
TRVSaveComponentToFileEvent 405
TRVSaveCSSOption 772
TRVSaveCSSOptions 772
TRVSaveDocXExtraEvent 407
TRVSaveFormat 1086
TRVSaveHTMLExtraEvent 409
TRVSaveImageEvent2 410
TRVSaveOption 1086
TRVSaveOptions 1086
TRVSaveParaToHTMLEvent 413
TRVSaveRTFExtraEvent 414
TRVScreenAndDevice 1090
TRVScroller 869
TRVScroller.BorderStyle 871
TRVScroller.Create 875
TRVScroller.HScrollMax 871
TRVScroller.HScrollPos 872

TRVScroller.HScrollVisible 872
TRVScroller.InplaceEditor 872
TRVScroller.NoVScroll 873
TRVScroller.OnHScrolled 877
TRVScroller.OnVScrolled 877
TRVScroller.ScrollTo 875
TRVScroller.Tracking 873
TRVScroller.UseXPThemes 873
TRVScroller.VScrollMax 874
TRVScroller.VScrollPos 874
TRVScroller.VScrollVisible 874
TRVScroller.WheelStep 875
TRVSearchOption 1091
TRVSearchOptions 1091
TRVSelection 1047
TRVSelectionHandleKind 757
TRVSelectionMode 758
TRVSelectionStyle 759
TRVSeqItemInfo 899, 978
TRVSeqItemInfo.Create 982
TRVSeqItemInfo.CreateEx 982
TRVSeqItemInfo.FormatString 980
TRVSeqItemInfo.NumberType 980
TRVSeqItemInfo.Reset 981
TRVSeqItemInfo.SeqName 981
TRVSeqItemInfo.StartFrom 981
TRVSeqType 1092
TRVSidenoteItemInfo.BoxPosition 993
TRVSidenoteItemInfo.BoxProperties 992
TRVSidenoteItemInfo.Create 993
TRVSidenoteItemInfo.CreateEx 993
TRVSmartPopupClickEvent 595
TRVSmartPopupPosition 601
TRVSmartPopupProperties 598
TRVSmartPopupProperties.ButtonType 599
TRVSmartPopupProperties.Color 599
TRVSmartPopupProperties.Hint 600
TRVSmartPopupProperties.HoverColor 600
TRVSmartPopupProperties.HoverLineColor 600
TRVSmartPopupProperties.ImageIndex 600
TRVSmartPopupProperties.ImageList 600
TRVSmartPopupProperties.LineColor 601
TRVSmartPopupProperties.Menu 601
TRVSmartPopupProperties.Popup 601
TRVSmartPopupProperties.Position 601
TRVSmartPopupProperties.SetButtonState 602
TRVSmartPopupProperties.ShortCut 601
TRVSpecialCharacter 1129
TRVSpecialCharacters 1129
TRVSpellChecker 728
TRVSpellChecker.AddToAutoreplaceList 733
TRVSpellChecker.AddToDictionary 733
TRVSpellChecker.AddToIgnoreList 733
TRVSpellChecker.AvailableLanguages 731
TRVSpellChecker.CurrentLanguage 733
TRVSpellChecker.DialogShown 731
TRVSpellChecker.Execute 733
TRVSpellChecker.FillLanguageNames 734
TRVSpellChecker.GetAutoCorrection 734
TRVSpellChecker.GetLanguageNameFromCode 735
TRVSpellChecker.GuessCompletions 735
TRVSpellChecker.IsAvailable 735
TRVSpellChecker.IsWordValid 735
TRVSpellChecker.Language 731
TRVSpellChecker.LanguageIndex 732
TRVSpellChecker.OnSpellForm 737
TRVSpellChecker.OnSpellFormAction 738
TRVSpellChecker.OnSpellFormLocalize 739
TRVSpellChecker.RegisterAsService 735
TRVSpellChecker.RegisterEditor 736
TRVSpellChecker.SpellFormStyle 732
TRVSpellChecker.StartLiveSpelling 736
TRVSpellChecker.Suggest 737
TRVSpellChecker.SupportsFeatures 737
TRVSpellChecker.UnregisterAsService 735
TRVSpellChecker.UnregisterEditor 736
TRVSpellCheckFormStyle 732
TRVSpellFormAction 1093
TRVSpellFormActions 1093
TRVSpellingCheckEvent 416
TRVSTFontInfo 847
TRVStickingItemsEvent 583
TRVSTParaInfo 846
TRVStringFormatMethod 1116
TRVStringList 1041
TRVStrProperty 1093
TRVStyle 739
TRVStyle.AddTextStyle 767
TRVStyle.CheckpointColor 743
TRVStyle.CheckpointEvColor 743
TRVStyle.Color 744
TRVStyle.ConvertToDifferentUnits 767
TRVStyle.ConvertToEMU 767
TRVStyle.ConvertToPixels 767
TRVStyle.ConvertToTwips 767
TRVStyle.Create 767
TRVStyle.CurrentItemColor 744
TRVStyle.DefCodePage 745

TRVStyle.DefTabWidth 745
TRVStyle.DefUnicodeStyle 746
TRVStyle.DeleteTextStyle 767
TRVStyle.Destroy 767
TRVStyle.DifferentUnitsToUnits 769
TRVStyle.DisabledFontColor 746
TRVStyle.EMUToUnits 769
TRVStyle.EndnoteNumbering 746
TRVStyle.FieldHighlightColor 746
TRVStyle.FieldHighlightType 747
TRVStyle.FindParaStyle 771
TRVStyle.FindTextStyle 771
TRVStyle.FloatingLineColor 747
TRVStyle.FontQuality 748
TRVStyle.FootnoteNumbering 746
TRVStyle.FootnotePageReset 749
TRVStyle.FullRedraw 750
TRVStyle.GetAsDifferentUnits 768
TRVStyle.GetAsEMU 768
TRVStyle.GetAsHTMLPixels 768
TRVStyle.GetAsPixels 768
TRVStyle.GetAsRVUnits 768
TRVStyle.GetAsStandardPixels 768
TRVStyle.GetAsTwips 768
TRVStyle.GetListStyleClass 770
TRVStyle.GetParaStyleClass 770
TRVStyle.GetTextStyleClass 770
TRVStyle.GridColor 750
TRVStyle.GridReadOnlyColor 750
TRVStyle.GridReadOnlyStyle 750
TRVStyle.GridStyle 750
TRVStyle.HoverColor 750
TRVStyle.HTMLPixelsToUnits 769
TRVStyle.InactiveSelColor 751
TRVStyle.InactiveSelTextColor 751
TRVStyle.InvalidPicture 752
TRVStyle.JumpCursor 752
TRVStyle.LineSelectCursor 753
TRVStyle.LineWrapMode 753
TRVStyle.ListStyles 755
TRVStyle.LiveSpellingColor 755
TRVStyle.LoadINI 771
TRVStyle.LoadReg 771
TRVStyle.MainRVStyle 756
TRVStyle.OnAfterApplyStyle 774
TRVStyle.OnApplyStyle 774
TRVStyle.OnApplyStyleColor 775
TRVStyle.OnDrawCheckpoint 776
TRVStyle.OnDrawPageBreak 778
TRVStyle.OnDrawParaBack 780
TRVStyle.OnDrawStyleText 781
TRVStyle.OnDrawTextBack 783
TRVStyle.OnStyleHoverSensitive 784
TRVStyle.PageBreakColor 756
TRVStyle.ParaBackAroundContent 757
TRVStyle.ParaStyles 757
TRVStyle.PixelsToUnits 769
TRVStyle.RVUnitsToUnits 769
TRVStyle.SaveCSS 772
TRVStyle.SaveCSSToStream 772
TRVStyle.SaveINI 773
TRVStyle.SaveReg 773
TRVStyle.SelColor 751
TRVStyle.SelectionHandleKind 757
TRVStyle.SelectionMode 758
TRVStyle.SelectionStyle 759
TRVStyle.SelOpacity 751
TRVStyle.SelTextColor 751
TRVStyle.SidenoteNumbering 746
TRVStyle.SoftPageBreakColor 761
TRVStyle.SpacesInTab 761
TRVStyle.SpecialCharactersColor 762
TRVStyle.StandardPixelsToUnits 769
TRVStyle.StyleTemplates 762
TRVStyle.TextBackgroundKind 762
TRVStyle.TextEngine 763
TRVStyle.TextStyles 764
TRVStyle.TwipsToUnits 769
TRVStyle.Units 764
TRVStyle.UnitsPixelsPerInch 765
TRVStyle.UseSound 766
TRVStyleConversionEvent 592, 595
TRVStyleHoverSensitiveEvent 784
TRVStyleLength 1094
TRVStyleTemplate 843
TRVStyleTemplate.ApplyToParaStyle 848
TRVStyleTemplate.ApplyToTextStyle 849
TRVStyleTemplate.Id 844
TRVStyleTemplate.Kind 845
TRVStyleTemplate.Name 845
TRVStyleTemplate.Next 846
TRVStyleTemplate.NextId 846
TRVStyleTemplate.ParaStyle 846
TRVStyleTemplate.Parent 847
TRVStyleTemplate.ParentId 847
TRVStyleTemplate.QuickAccess 847
TRVStyleTemplate.TextStyle 847

TRVStyleTemplate.UpdateModifiedParaStyleProperties 850
TRVStyleTemplate.UpdateModifiedTextStyleProperties 850
TRVStyleTemplate.ValidParaProperties 847
TRVStyleTemplate.ValidTextProperties 848
TRVStyleTemplateCollection 797
TRVStyleTemplateCollection.AssignStyleTemplates 799
TRVStyleTemplateCollection.AssignToStrings 799
TRVStyleTemplateCollection.ClearParaFormat 800
TRVStyleTemplateCollection.ClearTextFormat 800
TRVStyleTemplateCollection.DefStyleName 798
TRVStyleTemplateCollection.FindById 800
TRVStyleTemplateCollection.FindByName 800
TRVStyleTemplateCollection.FindItemById 800
TRVStyleTemplateCollection.FindItemByName 800
TRVStyleTemplateCollection.ForbiddenIds 798
TRVStyleTemplateCollection.InsertFromRVST 801
TRVStyleTemplateCollection.Items 798
TRVStyleTemplateCollection.LoadFromRVST 801
TRVStyleTemplateCollection.LoadFromStreamRVST 802
TRVStyleTemplateCollection.NormalStyleTemplate 799
TRVStyleTemplateCollection.ResetNameCounter 802
TRVStyleTemplateCollection.SaveToRVST 801
TRVStyleTemplateCollection.SaveToStreamRVST 802
TRVStyleTemplateId 1094
TRVStyleTemplateInsertMode 260
TRVStyleTemplateKind 845
TRVStyleTemplateName 1094
TRVStyleUnits 1094
TRVSubData 1014
TRVSubSuperScriptType 818
TRVTabAlign 856
TRVTabInfo 856
TRVTabInfo.Align 856
TRVTabInfo.Leader 857
TRVTabInfo.Position 857
TRVTabInfos 854
TRVTabInfos.AddFrom 855
TRVTabInfos.DeleteList 855
TRVTabInfos.Find 855
TRVTabInfos.Intersect 856
TRVTabInfos.Items 855
TRVTabItemInfo 899
TRVTableBorderStyle 1096
TRVTableCellData 951, 1014
TRVTableCellData.BackgroundImage 953
TRVTableCellData.BackgroundImageFileName 953
TRVTableCellData.BackgroundProperties 954
TRVTableCellData.BackgroundPropertiesAllocate 964
TRVTableCellData.BackgroundPropertiesFinalize 964
TRVTableCellData.BackgroundStyle 954
TRVTableCellData.BestHeight 955
TRVTableCellData.BestWidth 955
TRVTableCellData.BorderColor 956
TRVTableCellData.BorderLightColor 956
TRVTableCellData.Color 957
TRVTableCellData.ColSpan 957
TRVTableCellData.GetRVData 964
TRVTableCellData.Hint 958
TRVTableCellData.IgnoreContentHeight 958
TRVTableCellData.Opacity 958
TRVTableCellData.Options 959
TRVTableCellData.properties 965
TRVTableCellData.Rotation 961
TRVTableCellData.RowSpan 962
TRVTableCellData.Tag 963
TRVTableCellData.VAlign 963
TRVTableCellData.VisibleBorders 963
TRVTableCellOption 959
TRVTableCellOptions 959
TRVTableDrawBackEvent 949
TRVTableDrawBorderEvent 949
TRVTableItemInfo 899, 901
TRVTableItemInfo.AssignProperties 925
TRVTableItemInfo.BackgroundImage 904
TRVTableItemInfo.BackgroundImageFileName 905
TRVTableItemInfo.BackgroundProperties 905
TRVTableItemInfo.BackgroundStyle 905
TRVTableItemInfo.BestWidth 906
TRVTableItemInfo.BorderColor 906
TRVTableItemInfo.BorderHSpacing 907
TRVTableItemInfo.BorderLightColor 907
TRVTableItemInfo.BorderStyle 908
TRVTableItemInfo.BorderVSpacing 908
TRVTableItemInfo.BorderWidth 909
TRVTableItemInfo.CanMergeCells 925
TRVTableItemInfo.CanMergeSelectedCells 926
TRVTableItemInfo.CellBorderColor 909
TRVTableItemInfo.CellBorderLightColor 910
TRVTableItemInfo.CellBorderStyle 910
TRVTableItemInfo.CellBorderWidth 910
TRVTableItemInfo.CellHPadding 911
TRVTableItemInfo.CellHSpacing 911
TRVTableItemInfo.CellOverrideColor 912
TRVTableItemInfo.CellPadding 912
TRVTableItemInfo.Cells 912
TRVTableItemInfo.CellVPadding 913

TRVTableItemInfo.CellVSpacing 913
TRVTableItemInfo.Changed 927
TRVTableItemInfo.ColBandSize 913
TRVTableItemInfo.ColCount 914
TRVTableItemInfo.Color 914
TRVTableItemInfo.Create 927
TRVTableItemInfo.CreateEx 928
TRVTableItemInfo.DeleteCols 928
TRVTableItemInfo.DeleteEmptyCols 929
TRVTableItemInfo.DeleteEmptyRows 929
TRVTableItemInfo.DeleteRows 929
TRVTableItemInfo.DeleteSelectedCols 930
TRVTableItemInfo.DeleteSelectedRows 930
TRVTableItemInfo.Deselect 931
TRVTableItemInfo.Destroy 931
TRVTableItemInfo.EditCell 931
TRVTableItemInfo.EvenColumnsColor 923
TRVTableItemInfo.EvenRowsColor 923
TRVTableItemInfo.FirstColumnColor 923
TRVTableItemInfo.GetCellAt 932
TRVTableItemInfo.GetEditedCell 932
TRVTableItemInfo.GetNormalizedSelectionBounds 933
TRVTableItemInfo.GetSelectionBounds 933
TRVTableItemInfo.HeadingRowColor 923
TRVTableItemInfo.HeadingRowCount 914
TRVTableItemInfo.HOutermostRule 915
TRVTableItemInfo.HRuleColor 915
TRVTableItemInfo.HRuleWidth 916
TRVTableItemInfo.InsertCols 934
TRVTableItemInfo.InsertColsLeft 934
TRVTableItemInfo.InsertColsRight 935
TRVTableItemInfo.InsertRows 935
TRVTableItemInfo.InsertRowsAbove 936
TRVTableItemInfo.InsertRowsBelow 936
TRVTableItemInfo.IsCellSelected 936
TRVTableItemInfo.LastColumnColor 923
TRVTableItemInfo.LastRowColor 923
TRVTableItemInfo.LoadFromStream 937
TRVTableItemInfo.MergeCells 937
TRVTableItemInfo.MergeSelectedCells 938
TRVTableItemInfo.MoveRows 938
TRVTableItemInfo.OddColumnsColor 923
TRVTableItemInfo.OddRowsColor 923
TRVTableItemInfo.OnCellEditing 948
TRVTableItemInfo.OnDrawBackground 949
TRVTableItemInfo.OnDrawBorder 949
TRVTableItemInfo.Opacity 916
TRVTableItemInfo.Options 916
TRVTableItemInfo.PrintOptions 919
TRVTableItemInfo.RowBandSize 913
TRVTableItemInfo.RowCount 920
TRVTableItemInfo.Rows 920
TRVTableItemInfo.SaveRowsToStream 939
TRVTableItemInfo.SaveToStream 939
TRVTableItemInfo.Select 939
TRVTableItemInfo.SelectCols 940
TRVTableItemInfo.SelectRows 940
TRVTableItemInfo.SetBackgroundProperty 940
TRVTableItemInfo.SetCellBackgroundImage 941
TRVTableItemInfo.SetCellBackgroundImageFileName 941
TRVTableItemInfo.SetCellBackgroundProperty 941
TRVTableItemInfo.SetCellBackgroundStyle 942
TRVTableItemInfo.SetCellBestHeight 942
TRVTableItemInfo.SetCellBestWidth 942
TRVTableItemInfo.SetCellBorderColor 942
TRVTableItemInfo.SetCellBorderLightColor 943
TRVTableItemInfo.SetCellColor 943
TRVTableItemInfo.SetCellHint 943
TRVTableItemInfo.SetCellOpacity 943
TRVTableItemInfo.SetCellOptions 944
TRVTableItemInfo.SetCellRotation 944
TRVTableItemInfo.SetCellTag 944
TRVTableItemInfo.SetCellVAlign 945
TRVTableItemInfo.SetCellVisibleBorders 945
TRVTableItemInfo.SetRowKeepTogether 945
TRVTableItemInfo.SetRowPageBreakBefore 945
TRVTableItemInfo.SetRowVAlign 945
TRVTableItemInfo.SetTableVisibleBorders 946
TRVTableItemInfo.SplitSelectedCellsHorizontally 946
TRVTableItemInfo.SplitSelectedCellsVertically 946
TRVTableItemInfo.TextColSeparator 920
TRVTableItemInfo.TextRowSeparator 921
TRVTableItemInfo.UnmergeCells 947
TRVTableItemInfo.UnmergeSelectedCells 947
TRVTableItemInfo.VisibleBorders 921
TRVTableItemInfo.VOutermostRule 921
TRVTableItemInfo.VRuleColor 922
TRVTableItemInfo.VRuleWidth 922
TRVTableOption 916
TRVTableOptions 916
TRVTablePrintOption 919
TRVTablePrintOptions 919
TRVTableRow 966
TRVTableRow.Items 967
TRVTableRow.KeepTogether 967
TRVTableRow.PageBreakBefore 967
TRVTableRow.VAlign 968

TRVTableRows 968
TRVTableRows.GetMainCell 969
TRVTableRows.Items 969
TRVTabNavigationType 262
TRVTag 1096
TRVTextBackgroundKind 762
TRVTextBoxItemInfo 1004
TRVTextBoxItemInfo.Create 1005
TRVTextBoxItemInfo.CreateEx 1005
TRVTextDrawState 1097
TRVTextDrawStates 1097
TRVTextEngine 763
TRVTextFoundEvent 418
TRVTextItemInfo 899
TRVTextOption 814
TRVTextOptions 814
TRVThumbnailMaker 1041
TRVTInplaceRVData 1014
TRVUnderlineType 819
TRVUndoType 1098
TRVUnicodeString 1099
TRVUnicodeTestResult 1054
TRVUnits 1099
TRVUnitsRect 1042
TRVVAlign 1100
TRVVerticalAnchor 1002
TRVVirtualPrinterProperties 720
TRVVirtualPrinterProperties.Active 721
TRVVirtualPrinterProperties.PageHeight 721
TRVVirtualPrinterProperties.PageWidth 721
TRVWriteHyperlink 418
TRVWriteObjectPropertiesEvent 420
TRVYesNoAuto 1101
TRVZoomMode 895
TRVZoomPercent 1102
TwipsToUnits 769
  TRVStyle 769

- U -

UnderlineColor 818
  TCustomRVFontInfo 818
UnderlineType 819
  TCustomRVFontInfo 819
Undo 577
  TRichViewEdit 577
Undo in RichViewEdit 208
Undo in TRichViewEdit 120
Undo/redo in Tables 208
UndoAction 577
  TRichViewEdit 577
UndoLimit 490
  TRichViewEdit 490
UndoName 578
  TRichViewEdit 578
Unicode 135, 824
  TFontInfo 824
Unicode in TRichView 135
UnicodeMode 467
  TRVRTFReaderProperties 467
Units 428, 707, 764
  TRVDocParameters 428
  TRVPrint 707
  TRVStyle 764
Units of measurement 144
UnitsPerInch 429
  TRVDocParameters 429
UnitsPixelsPerInch 765
  TRVStyle 765
UnmergeCells 947
  TRVTableItemInfo 947
UnmergeSelectedCells 947
  TRVTableItemInfo 947
UnregisterAsService 735
  TRVSpellChecker 735
UnregisterEditor 736
  TRVSpellChecker 736
UpdateModifiedParaStyleProperties 850
  TRVStyleTemplate 850
UpdateModifiedTextStyleProperties 850
  TRVStyleTemplate 850
UpdatePageNumbers 727
  TRVReportHelper 727
UpdatePaletteInfo 375, 884
  TCustomRVPrint 884
  TRichView 375
UpdateParaStyles 654
  TRVChat 654
UpdateTextStyles 654
  TRVChat 654
URLFormatName 1104
UseCharsetForUnicode 468
  TRVRTFReaderProperties 468
UseCheckpointNames 445
  TRVHTMLSaveProperties 445
UseFMXThemes 263
  TRichView 263
UseHypertextStyles 468
  TRVRTFReaderProperties 468

User 667, 674, 676
TRVChatMessage 667
TRVChatMessageExtraQuote 674
TRVChatMessageQuote 676
UserNameFont 664
TRVChatFonts 664
UserPicture 644
TRVChat 644
Users 644
TRVChat 644
UseSingleCellPadding 468
TRVRTFReaderProperties 468
UseSound 766
TRVStyle 766
UseStyleTemplates 263, 1039
TRichView 263
TRVReportHelperWithHeaderFooters 1039
UseVCLThemes 264
TRichView 264
UseXPThemes 873
TRVScroller 873

- V -

Valid documents 143
Valid TRichView documents 143
ValidParaProperties 847
TRVStyleTemplate 847
ValidTextProperties 848
TRVStyleTemplate 848
VAlign 264, 963, 968, 996
TRichView 264
TRVBoxProperties 996
TRVTableCellData 963
TRVTableRow 968
VerticalAlignment 1003, 1024
TRVBackgroundProperties 1024
TRVBoxPosition 1003
VerticalAnchor 1002
TRVBoxPosition 1002
VerticalizeKind 1025
VerticallOffset 1025
VerticalOffset 1003
TRVBackgroundProperties 1025
TRVBoxPosition 1003
VerticalOffsetKind 1025
TRVBackgroundProperties 1025
VerticalPositionKind 1003
TRVBoxPosition 1003
VerticalRepeat 1025
TRVBackgroundProperties 1025
VerticalSizeKind
TRVBackgroundProperties 1025
Viewer vs Editor 140
VirtualPrinter 707
TRVPrint 707
Visible 1025
TRVBackgroundProperties 1025
VisibleBorders 854, 921, 963
TRVBorder 854
TRVTableCellData 963
TRVTableItemInfo 921
VOutermostRule 921
TRVTableItemInfo 921
VRuleColor 922
TRVTableItemInfo 922
VRuleWidth 922
TRVTableItemInfo 922
VScrollMax 874
TRVScroller 874
VScrollPos 874
TRVScroller 874
VScrollVisible 874
TRVScroller 874
VShift 820
TCustomRVFontInfo 820
VSmallStep 264
TRichView 264

- W -

WheelStep 875
TRVScroller 875
Width 854, 996
TRVBorder 854
TRVBoxProperties 996
WidthType 996
TRVBoxProperties 996
WordWrap 265
TRichView 265

- X -

XHTMLCheck 436
TRVHTMLReaderProperties 436

- Z -

ZoomInCursor 895
TCustomRVPrintPreview 895

ZoomMode 428, 895
TCustomRVPrintPreview 895
TRVDocParameters 428
ZoomOutCursor 896
TCustomRVPrintPreview 896
ZoomPercent 265, 428, 896
TCustomRVPrintPreview 896
TRichView 265
TRVDocParameters 428