



TRichView Components

TRichView © trichview.com

Table of Contents

Part I TRichView reference	32
Part II Version history	34
1..New in version 23	34
2..New in version 22	36
3..New in version 21	38
4..New in version 20	40
5..New in version 19	42
6..New in version 18	43
7..New in version 17	46
8..New in version 16	48
9..New in version 15	51
10..New in version 14	53
11..New in version 13	57
12..New in version 12	59
13..New in version 11	60
14..New in version 10	63
15..New in version 1.9	67
16..New in version 1.8	70
17..New in version 1.7	72
18..New in version 1.6	75
19..New in version 1.5	77
20..New in version 1.4	80
21..New in version 1.3	81
22..New in version 1.2	82
Part III Overview	84
1..Parts of documents	85
TRichView item types	85
"Checkpoints"	87
Items' "tags"	91
Hypertext	92
Paragraphs	95
Borders and background of paragraphs	97
Styles and styleTemplates	98
Hidden text	101
2..Basic Operations	101
Building TRichView document	102
Scrolling	105
Selecting	107
Clipboard	108
Searching and replacing	110

Obtaining Items	111
Modifying Items	112
3. Editing	114
Inserting at the position of caret	114
Undo and redo	115
Drag and drop	118
4. Graphics	119
Animated images	119
Smooth image scaling	120
Semitransparent objects	121
5. Saving and loading	122
Saving and loading	122
RVF (RichView Format)	124
Export to HTML	127
6. Internationalization	130
Unicode	130
Bidirectional text	132
7. Spelling check	132
Live spelling check	132
8. Technical information	134
Viewer vs editor	135
Controls - documents - items	136
Valid documents	138
Units of measurement	139
Custom drawing	141
Cursors	142
RVF specification	143
Lazarus	154
FireMonkey	155
Skia and Skia4Delphi	157
Part IV Overview of Items	158
1. Overview	158
2. Text items	161
3. Tabulators	162
4. Pictures	163
5. Hot-pictures	166
6. Breaks	167
7. Controls	168
8. Bullets	170
9. Hotspots	172
10. Tables	173
11. List markers	174
12. Labels	176
13. Numbered sequences	177
14. Endnotes	178
15. Footnotes	180
16. Sidenotes	181
17. Text boxes	183

18..References to notes	184
19..Page numbers	185
20..Page counts	186
21..Mathematical expressions	187
22..Custom item types	189

Part V Overview of tables 189

1..Tables in TRichView	190
2..Addressing items in table cells	191
3..Table cells	192
4..Colors and layout of tables	193
5..Selection in table	195
6..Editing cells	197
7..Cell merging	198
8..Table operations	199
9..Undo/redo in tables	202
10..Export of tables	203
11..Table resizing	205
12..Keys in editor	206
13..Example: table to text	206
14..Example: dividing table	208

Part VI Components 209

1..Visual Components	210
TRichView	210
TRichView: component editor	218
Properties	219
TRichView.AllowSelection	222
TRichView.AnimationMode.....	222
TRichView.BackgroundImage.....	222
TRichView.BackgroundColor.....	223
TRichView.BiDiMode	224
TRichView.BottomMargin.....	225
TRichView.ClearLeft	225
TRichView.ClearRight	226
TRichView.Color	226
TRichView.CPEventKind	227
TRichView.Cursor	227
TRichView.DarkMode	228
TRichView.Delimiters	228
TRichView.DocObjects	229
TRichView.DocParameters	230
TRichView.DocProperties	231
TRichView.Document	232
TRichView.DocumentHeight.....	233
TRichView.DocumentPixelsPerInch.....	233
TRichView.DoInPaletteMode.....	234
TRichView.FirstItemVisible.....	236
TRichView.FirstJumpNo	236
TRichView.HTMLReadProperties	237

TRichView.HTMLSaveProperties.....	237
TRichView.ItemCount	237
TRichView.LastItemVisible.....	238
TRichView.LeftMargin	238
TRichView.LineCount	238
TRichView.MarkdownProperties.....	239
TRichView.MaxLength	239
TRichView.MaxTextWidth.....	239
TRichView.MinTextWidth.....	240
TRichView.NoteText	240
TRichView.Options	240
TRichView.PageBreaksBeforeItems	244
TRichView.RightMargin	244
TRichView.RTFOptions	245
TRichView.RTFReadProperties.....	246
TRichView.RVData	247
TRichView.RVFOptions	247
TRichView.RVFParaStylesReadMode.....	251
TRichView.RVFTextStylesReadMode.....	251
TRichView.RVFWarnings.....	251
TRichView.SelectionHandlesVisible.....	253
TRichView.SingleClick	253
TRichView.Style	253
TRichView.StyleTemplatInsertMode.....	254
TRichView.TabNavigation.....	255
TRichView.TopMargin	255
TRichView.UseFMXThemes.....	256
TRichView.UseStyleTemplates	256
TRichView.UseVCLThemes.....	257
TRichView.VAlign	257
TRichView.VSmallStep	257
TRichView.WordWrap	258
TRichView.ZoomPercent.....	258
Methods	258
TRichView.AddBreak	262
TRichView.AddBullet	263
TRichView.AddCheckpoint.....	264
TRichView.AddControl	265
TRichView.AddFmt	266
TRichView.AddHotPicture.....	267
TRichView.AddHotspot	268
TRichView.AddItem	269
TRichView.AddNL -A -W	270
TRichView.AddPicture	272
TRichView.AddTab	273
TRichView.AddTextNL -A -W.....	274
TRichView.AppendFrom.....	275
TRichView.AppendRVFFromStream.....	276
TRichView.AssignSoftPageBreaks	277
TRichView.BeginOleDrag.....	278
TRichView.Clear	279
TRichView.ClearLiveSpellingResults.....	279
TRichView.ClearSoftPageBreaks.....	279
TRichView.ClientToDocument, DocumentToClient.....	280
TRichView.ConvertDocTo.....	280
TRichView.Copy	281
TRichView.CopyDef	281
TRichView.CopyImage	281

TRichView.CopyRTF	282
TRichView.CopyRVF	282
TRichView.CopyText -W	283
TRichView.Create	283
TRichView.DeleteItems	283
TRichView.DeleteMarkedStyles	284
TRichView.DeleteParas	285
TRichView.DeleteSection.....	285
TRichView.DeleteUnusedStyles.....	286
TRichView.Deselect	286
TRichView.Destroy	286
TRichView.FindCheckpointByName.....	287
TRichView.FindCheckpointByTag.....	287
TRichView.FindControlItemNo.....	287
TRichView.Format	288
TRichView.FormatTail	288
TRichView.GetBreakInfo.....	289
TRichView.GetBulletInfo.....	289
TRichView.GetCheckpointByNo.....	291
TRichView.GetCheckpointInfo.....	291
TRichView.GetCheckpointItemNo.....	291
TRichView.GetCheckpointNo.....	292
TRichView.GetCheckpointXY.....	292
TRichView.GetCheckpointY.....	292
TRichView.GetCheckpointYEx.....	293
TRichView.GetControlInfo.....	293
TRichView.GetFirstCheckpoint.....	294
TRichView.GetFocusedItem.....	295
TRichView.GetHotspotInfo.....	295
TRichView.GetItem	296
TRichView.GetItemAt	297
TRichView.GetItemCheckpoint.....	298
TRichView.GetItemCoords	298
TRichView.GetItemCoordsEx	299
TRichView.GetItemExtraIntProperty -Ex.....	300
TRichView.GetItemExtraStrProperty -Ex.....	300
TRichView.GetItemNo	301
TRichView.GetItemPara	301
TRichView.GetItemStyle	302
TRichView.GetItemTag	302
TRichView.GetItemText -A -W.....	303
TRichView.GetItemVAlign.....	303
TRichView.GetJumpPointItemNo.....	304
TRichView.GetJumpPointLocation.....	304
TRichView.GetJumpPointY.....	305
TRichView.GetLastCheckpoint.....	305
TRichView.GetLineNo	306
TRichView.GetListMarkerInfo.....	307
TRichView.GetNextCheckpoint.....	307
TRichView.GetOffsAfterItem.....	308
TRichView.GetOffsBeforeItem.....	309
TRichView.GetPictureInfo.....	310
TRichView.GetPrevCheckpoint.....	311
TRichView.GetRealDocumentPixelsPerInch.....	311
TRichView.GetSelectedImage.....	312
TRichView.GetSelectionBounds.....	312
TRichView.GetSelText -A -W.....	313
TRichView.GetTextInfo	313

TRichView.GetWordAt -A -W	314
TRichView.InsertRVFFromStream.....	316
TRichView.IsFromNewLine.....	317
TRichView.IsParaStart	318
TRichView.LiveSpellingValidateWord.....	318
TRichView.LoadDocX	318
TRichView.LoadDocXFromStream.....	319
TRichView.LoadFromFile -Ex	320
TRichView.LoadFromStream -Ex.....	321
TRichView.LoadHTML	322
TRichView.LoadHTMLFromStream.....	324
TRichView.LoadMarkdown.....	325
TRichView.LoadMarkdownFromStream.....	325
TRichView.LoadRTF	326
TRichView.LoadRTFFromStream.....	327
TRichView.LoadRVF	328
TRichView.LoadRVFFromStream.....	329
TRichView.LoadText -W	330
TRichView.LoadTextFromStream -W.....	331
TRichView.MarkStylesInUse.....	331
TRichView.Reformat	332
TRichView.RefreshListMarkers.....	332
TRichView.RefreshListSequences	332
TRichView.RemoveCheckpoint.....	332
TRichView.ResetAnimation.....	333
TRichView.SaveDocX	333
TRichView.SaveDocXToStream.....	334
TRichView.SaveHTML	335
TRichView.SaveHTML (deprecated).....	336
TRichView.SaveHTMLEx (deprecated).....	337
TRichView.SaveHTMLToStream.....	338
TRichView.SaveHTMLToStream (deprecated).....	340
TRichView.SaveHTMLToStreamEx (deprecated).....	340
TRichView.SaveMarkdown	340
TRichView.SaveMarkdownToStream.....	341
TRichView.SavePicture	342
TRichView.SaveRTF	343
TRichView.SaveRTFToStream.....	344
TRichView.SaveRVF	344
TRichView.SaveRVFToStream.....	344
TRichView.SaveText -W	345
TRichView.SaveTextToStream -W.....	345
TRichView.SearchText -A -W.....	346
TRichView.SelectAll	348
TRichView.SelectControl.....	348
TRichView.SelectionExists.....	349
TRichView.SelectWordAt.....	349
TRichView.SetAddParagraphMode.....	350
TRichView.SetBreakInfo.....	351
TRichView.SetBulletInfo.....	352
TRichView.SetCheckpointInfo.....	353
TRichView.SetControlInfo.....	354
TRichView.SetFooter	355
TRichView.SetHeader	356
TRichView.SetHotspotInfo.....	357
TRichView.SetItemExtraIntProperty -Ex.....	358
TRichView.SetItemExtraStrProperty -Ex.....	359
TRichView.SetItemTag	360

TRichView.SetItemText -A -W.....	360
TRichView.SetItemVAlign.....	361
TRichView.SetListMarkerInfo.....	362
TRichView.SetPictureInfo.....	363
TRichView.SetSelectionBounds.....	365
TRichView.StartAnimation.....	366
TRichView.StartLiveSpelling.....	367
TRichView.StopAnimation.....	367
TRichView.StoreSearchResult.....	367
TRichView.UpdatePalettelInfo.....	368
Events	368
TRichView.OnAddStyle	370
TRichView.OnAfterDrawImage.....	370
TRichView.OnAssignImageFileName.....	371
TRichView.OnCheckpointVisible.....	372
TRichView.OnCMHintShow.....	372
TRichView.OnControlAction.....	373
TRichView.OnCopy	374
TRichView.OnGetItemCursor.....	374
TRichView.OnGetSpellingSuggestions.....	375
TRichView.OnHTMLSaveImage.....	375
TRichView.OnImportFile.....	378
TRichView.OnImportPicture.....	378
TRichView.OnItemAction.....	380
TRichView.OnItemHint	381
TRichView.OnJump	382
TRichView.OnLoadCustomFormat.....	383
TRichView.OnLoadDocument.....	384
TRichView.OnNewDocument.....	384
TRichView.OnPaint	384
TRichView.OnProgress	385
TRichView.OnReadField.....	386
TRichView.OnReadHyperlink.....	388
TRichView.OnReadMergeField.....	390
TRichView.OnRVDbClick.....	391
TRichView.OnRVFControlNeeded.....	392
TRichView.OnRVFImageListNeeded.....	393
TRichView.OnRVFPictureNeeded.....	393
TRichView.OnRVMouseDown.....	394
TRichView.OnRVMouseMove.....	395
TRichView.OnRVMouseUp.....	396
TRichView.OnRVRightClick.....	397
TRichView.OnSaveComponentToFile.....	397
TRichView.OnSaveDocXExtra.....	399
TRichView.OnSaveHTMLExtra.....	401
TRichView.OnSaveImage2.....	402
TRichView.OnSaveItemToFile.....	404
TRichView.OnSaveParaToHTML.....	406
TRichView.OnSaveRTFExtra.....	406
TRichView.OnSelect	408
TRichView.OnSpellingCheck.....	409
TRichView.OnStyleTemplatesChange.....	410
TRichView.OnTextFound.....	410
TRichView.OnWriteHyperlink.....	411
TRichView.OnWriteObjectProperties.....	413
Classes of properties.....	415
TRVDocParameters	415
Properties	416

TRVDocParameters.Author.....	416
TRVDocParameters.BottomMargin.....	416
TRVDocParameters.Comments.....	417
TRVDocParameters.FacingPages.....	417
TRVDocParameters.FooterY.....	417
TRVDocParameters.HeaderY.....	418
TRVDocParameters.LeftMargin.....	418
TRVDocParameters.MirrorMargins.....	418
TRVDocParameters.Orientation.....	418
TRVDocParameters.PageHeight.....	419
TRVDocParameters.PageWidth.....	419
TRVDocParameters.RightMargin.....	419
TRVDocParameters.Title.....	419
TRVDocParameters.TitlePage.....	419
TRVDocParameters.TopMargin.....	420
TRVDocParameters.Units.....	420
TRVDocParameters.ZoomMode.....	420
TRVDocParameters.ZoomPercent.....	421
Methods.....	421
TRVDocParameters.ConvertToUnits.....	421
TRVDocParameters.Reset.....	421
TRVDocParameters.ResetLayout.....	421
TRVDocParameters.UnitsPerInch.....	421
TRVDocumentProperty.....	422
Properties.....	422
TRVDocumentProperty.AllowMarkdown.....	422
TRVDocumentProperty.AutoDeleteUnusedStyles.....	423
TRVDocumentProperty.CodePage.....	423
TRVDocumentProperty.FieldFormat.....	424
TRVDocumentProperty.IgnoreEscape.....	424
TRVHTMLReaderProperties.....	424
Properties.....	425
TRVHTMLReaderProperties.BasePathLinks.....	425
TRVHTMLReaderProperties.EnforceListLevels.....	426
TRVHTMLReaderProperties.FontSizePrecision.....	426
TRVHTMLReaderProperties.IDAsCheckpoints.....	427
TRVHTMLReaderProperties.ReadBackground.....	427
TRVHTMLReaderProperties.ReadDocParameters.....	427
TRVHTMLReaderProperties.SkipHiddenText.....	428
TRVHTMLReaderProperties.XHTMLCheck.....	428
TRVHTMLSaveProperties.....	429
Properties.....	429
TRVHTMLSaveProperties.CheckpointsPrefix.....	430
TRVHTMLSaveProperties.CSSOptions.....	430
TRVHTMLSaveProperties.CSSSavingType.....	431
TRVHTMLSaveProperties.Default0Style.....	431
TRVHTMLSaveProperties.Encoding.....	431
TRVHTMLSaveProperties.EncodingStr.....	432
TRVHTMLSaveProperties.ExternalCSSFileName.....	432
TRVHTMLSaveProperties.ExtraStyles.....	433
TRVHTMLSaveProperties.HTMLSavingType.....	433
TRVHTMLSaveProperties.HTMLVersion.....	433
TRVHTMLSaveProperties.ImageOptions.....	434
TRVHTMLSaveProperties.ImagesPrefix.....	436
TRVHTMLSaveProperties.ListMarkerSavingType.....	436
TRVHTMLSaveProperties.SaveHeaderAndFooter.....	437
TRVHTMLSaveProperties.UseCheckpointNames.....	437
TRVMarkdownDefaultItemProperties.....	438

Properties	438
TRVMarkdownDefaultItemProperties.BlockQuoteBackColor	438
TRVMarkdownDefaultItemProperties.BreakColor	438
TRVMarkdownDefaultItemProperties.CodeBlockBackColor	439
TRVMarkdownDefaultItemProperties.ImageVAlign	439
TRVMarkdownDefaultItemProperties.ListMarkerIndentPix, ListLeftIndentPix	439
TRVMarkdownDefaultItemProperties.TableCellWidthPix	439
TRVMarkdownProperties	439
Properties	444
TRVMarkdownProperties.DefaultItemProperties	444
TRVMarkdownProperties.Extensions	444
TRVMarkdownProperties.ImagesPrefix	445
TRVMarkdownProperties.LineWidth	445
TRVMarkdownProperties.LoadOptions	446
TRVMarkdownProperties.NotesSavedAsFootnotes	448
TRVMarkdownProperties.SaveHiddenText	448
TRVMarkdownProperties.SaveOptions	448
TRVRTFReaderProperties	450
Properties	450
TRVRTFReaderProperties.AutoHideTableGridLines	451
TRVRTFReaderProperties.BasePathLinks	451
TRVRTFReaderProperties.CanReuseNumberedLists	452
TRVRTFReaderProperties.CharsetForUnicode	452
TRVRTFReaderProperties.ConvertHighlight	452
TRVRTFReaderProperties.ConvertSymbolFonts	453
TRVRTFReaderProperties.DocXErrorCode	454
TRVRTFReaderProperties.ExtractMetafileBitmaps	454
TRVRTFReaderProperties.IgnoreBookmarks	454
TRVRTFReaderProperties.IgnoreFields	455
TRVRTFReaderProperties.IgnoreNotes	455
TRVRTFReaderProperties.IgnorePictures	455
TRVRTFReaderProperties.IgnoreSequences	455
TRVRTFReaderProperties.IgnoreTables	455
TRVRTFReaderProperties.LineBreaksAsParagraphs	456
TRVRTFReaderProperties.ParaStyleMode	456
TRVRTFReaderProperties.ParaStyleNo	457
TRVRTFReaderProperties.ReadDocParameters	457
TRVRTFReaderProperties.ReadNoteNumberingType	457
TRVRTFReaderProperties.RTFErrorCode	457
TRVRTFReaderProperties.SkipHiddenText	458
TRVRTFReaderProperties.TextStyleMode	458
TRVRTFReaderProperties.TextStyleNo	459
TRVRTFReaderProperties.UnicodeMode	459
TRVRTFReaderProperties.UseCharsetForUnicode	459
TRVRTFReaderProperties.UseHypertextStyles	459
TRVRTFReaderProperties.UseSingleCellPadding	460
Examples	460
Loading UTF-8 files	460
TRichViewEdit	461
Properties	467
TRichViewEdit.AcceptDragDropFormats, AcceptPasteFormats	470
TRichViewEdit.ActualCurTextStyleNo	472
TRichViewEdit.CheckSpelling	472
TRichViewEdit.CurlItemNo	472
TRichViewEdit.CurlItemStyle	473
TRichViewEdit.CurParaStyleNo	473
TRichViewEdit.CurTextStyleNo	474
TRichViewEdit.CustomCaretInterval	474

TRichViewEdit.DefaultPictureVAlign	475
TRichViewEdit.EditorOptions	475
TRichViewEdit.ForceFieldHighlight	478
TRichViewEdit.KeyboardType	478
TRichViewEdit.LiveSpellingMode	479
TRichViewEdit.Modified	479
TRichViewEdit.OffsetInCurlItem	479
TRichViewEdit.ReadOnly	480
TRichViewEdit.SmartPopupProperties	480
TRichViewEdit.SmartPopupVisible	481
TRichViewEdit.TopLevelEditor	481
TRichViewEdit.UndoLimit	481
Methods	482
TRichViewEdit.AddSpellingMenuItems	488
TRichViewEdit.AdjustControlPlacement	489
TRichViewEdit.AdjustControlPlacement2	489
TRichViewEdit.ApplyListStyle	490
TRichViewEdit.ApplyParaStyle	490
TRichViewEdit.ApplyParaStyleConversion	491
TRichViewEdit.ApplyParaStyleTemplate	491
TRichViewEdit.ApplyStyleConversion	492
TRichViewEdit.ApplyStyleTemplate	493
TRichViewEdit.ApplyTextStyle	494
TRichViewEdit.ApplyTextStyleTemplate	494
TRichViewEdit.BeginItemModify	495
TRichViewEdit.BeginUndoCustomGroup	496
TRichViewEdit.BeginUndoCustomGroup2, EndUndoCustomGroup2	496
TRichViewEdit.BeginUndoGroup	497
TRichViewEdit.BeginUndoGroup2, EndUndoGroup2	497
TRichViewEdit.BeginUpdate	498
TRichViewEdit.CanChange	498
TRichViewEdit.CanPaste	498
TRichViewEdit.CanPasteHTML	499
TRichViewEdit.CanPasteRTF	499
TRichViewEdit.CanPasteRVF	499
TRichViewEdit.Change	499
TRichViewEdit.ChangeListLevels	500
TRichViewEdit.ChangeStyleTemplates	500
TRichViewEdit.Clear	501
TRichViewEdit.ClearTextFlow	501
TRichViewEdit.ClearUndo	501
TRichViewEdit.ConvertToHotPicture	501
TRichViewEdit.ConvertToPicture	502
TRichViewEdit.Create	502
TRichViewEdit.CutDef	502
TRichViewEdit.DeleteSelection	503
TRichViewEdit.Destroy	503
TRichViewEdit.EndItemModify	503
TRichViewEdit.EndUpdate	503
TRichViewEdit.GetCheckpointAtCaret	503
TRichViewEdit.GetCurrentBreakInfo	504
TRichViewEdit.GetCurrentBulletInfo	505
TRichViewEdit.GetCurrentCheckpoint	506
TRichViewEdit.GetCurrentControlInfo	506
TRichViewEdit.GetCurrentHotspotInfo	507
TRichViewEdit.GetCurrentItem	508
TRichViewEdit.GetCurrentItemEx	509
TRichViewEdit.GetCurrentItemExtraIntProperty -Ex	510

TRichViewEdit.GetCurrentItemExtraStrProperty -Ex	510
TRichViewEdit.GetCurrentItemText -A -W	511
TRichViewEdit.GetCurrentItemVAlign	512
TRichViewEdit.GetCurrentLineCol	512
TRichViewEdit.GetCurrentMisspelling	513
TRichViewEdit.GetCurrentPictureInfo	513
TRichViewEdit.GetCurrentTag	514
TRichViewEdit.GetCurrentTextInfo	514
TRichViewEdit.InsertBreak	515
TRichViewEdit.InsertBullet	515
TRichViewEdit.InsertCheckpoint	516
TRichViewEdit.InsertControl	517
TRichViewEdit.InsertDocXFromFileEd	518
TRichViewEdit.InsertDocXFromStreamEd	518
TRichViewEdit.InsertHotPicture	519
TRichViewEdit.InsertHotspot	520
TRichViewEdit.InsertHTMLFromFileEd	521
TRichViewEdit.InsertHTMLFromStreamEd	521
TRichViewEdit.InsertItem	522
TRichViewEdit.InsertMarkdownFromFileEd	523
TRichViewEdit.InsertMarkdownFromStreamEd	524
TRichViewEdit.InsertOEMTextFromFile	525
TRichViewEdit.InsertPageBreak	525
TRichViewEdit.InsertPicture	526
TRichViewEdit.InsertRTFFFromFileEd	527
TRichViewEdit.InsertRTFFFromStreamEd	528
TRichViewEdit.InsertRVFFFromFileEd	528
TRichViewEdit.InsertRVFFFromStreamEd	529
TRichViewEdit.InsertStringTag -A -W	530
TRichViewEdit.InsertTab	531
TRichViewEdit.InsertText -A -W	532
TRichViewEdit.InsertTextFromFile -W	533
TRichViewEdit.MoveCaret	534
TRichViewEdit.Paste	534
TRichViewEdit.PasteBitmap	535
TRichViewEdit.PasteGraphicFiles	535
TRichViewEdit.PasteHTML	536
TRichViewEdit.PasteMetafile	537
TRichViewEdit.PasteRTF	537
TRichViewEdit.PasteRVF	538
TRichViewEdit.PasteText -A -W	538
TRichViewEdit.PasteURL	539
TRichViewEdit.Redo	540
TRichViewEdit.RedoAction	540
TRichViewEdit.RedoName	540
TRichViewEdit.RemoveCheckpointAtCaret	541
TRichViewEdit.RemoveCheckpointEd	541
TRichViewEdit.RemoveCurrentCheckpoint	541
TRichViewEdit.RemoveCurrentPageBreak	542
TRichViewEdit.RemoveLists	542
TRichViewEdit.ResizeControl	542
TRichViewEdit.ResizeCurrentControl	543
TRichViewEdit.SearchText -W	543
TRichViewEdit.SelectCurrentLine	545
TRichViewEdit.SelectCurrentWord	545
TRichViewEdit.Set*PropertyEd	545
TRichViewEdit.SetBackgroundImageEd	546
TRichViewEdit.SetBreakInfoEd	547

TRichViewEdit.SetBulletInfoEd.....	548
TRichViewEdit.SetCheckpointInfoEd.....	549
TRichViewEdit.SetControlInfoEd.....	550
TRichViewEdit.SetCurrentBreakInfo.....	551
TRichViewEdit.SetCurrentBulletInfo.....	552
TRichViewEdit.SetCurrentCheckpointInfo.....	553
TRichViewEdit.SetCurrentControlInfo.....	554
TRichViewEdit.SetCurrentHotspotInfo.....	555
TRichViewEdit.SetCurrentItemExtraIntProperty -Ex.....	556
TRichViewEdit.SetCurrentItemExtraStrProperty -Ex.....	557
TRichViewEdit.SetCurrentItemText -A -W.....	557
TRichViewEdit.SetCurrentItemVAlign.....	558
TRichViewEdit.SetCurrentPictureInfo.....	559
TRichViewEdit.SetCurrentTag.....	560
TRichViewEdit.SetHotspotInfoEd.....	560
TRichViewEdit.SetItemExtraIntProperty-Ed -ExEd.....	562
TRichViewEdit.SetItemExtraStrProperty-Ed -ExEd.....	562
TRichViewEdit.SetItemTagEd.....	563
TRichViewEdit.SetItemTextEd -A -W.....	564
TRichViewEdit.SetItemVAlignEd.....	565
TRichViewEdit.SetPictureInfoEd.....	565
TRichViewEdit.SetUndoGroupMode.....	567
TRichViewEdit.Undo.....	567
TRichViewEdit.UndoAction.....	567
TRichViewEdit.UndoName.....	568
Events.....	568
TRichViewEdit.OnBeforeOleDrop, OnAfterOleDrop.....	570
TRichViewEdit.OnCaretGetOut.....	571
TRichViewEdit.OnCaretMove.....	572
TRichViewEdit.OnChange.....	572
TRichViewEdit.OnChanging.....	572
TRichViewEdit.OnCheckStickingItems.....	573
TRichViewEdit.OnCurParaStyleChanged.....	573
TRichViewEdit.OnCurTextStyleChanged.....	574
TRichViewEdit.OnDrawCurrentLine.....	574
TRichViewEdit.OnDrawCustomCaret.....	575
TRichViewEdit.OnDropFile.....	576
TRichViewEdit.OnDropFiles.....	576
TRichViewEdit.OnItemResize.....	577
TRichViewEdit.OnItemTextEdit.....	578
TRichViewEdit.OnMeasureCustomCaret.....	578
TRichViewEdit.OnOleDragEnter.....	579
TRichViewEdit.OnOleDragLeave.....	579
TRichViewEdit.OnOleDragOver.....	580
TRichViewEdit.OnOleDrop.....	581
TRichViewEdit.OnParaStyleConversion.....	582
TRichViewEdit.OnPaste.....	584
TRichViewEdit.OnSaveCustomFormat.....	585
TRichViewEdit.OnSmartPopupClick.....	585
TRichViewEdit.OnStyleConversion.....	585
Classes of properties.....	588
TRVSmartPopupProperties.....	588
Properties.....	588
TRVSmartPopupProperties.ButtonType.....	589
TRVSmartPopupProperties.Color.....	589
TRVSmartPopupProperties.Hint.....	590
TRVSmartPopupProperties.HoverColor.....	590
TRVSmartPopupProperties.HoverLineColor.....	590

TRVSmartPopupProperties.ImageIndex.....	590
TRVSmartPopupProperties.ImageList.....	590
TRVSmartPopupProperties.LineColor.....	591
TRVSmartPopupProperties.Menu.....	591
TRVSmartPopupProperties.Popup.....	591
TRVSmartPopupProperties.Position.....	591
TRVSmartPopupProperties.ShortCut.....	591
Methods.....	592
TRVSmartPopupProperties.SetButtonState.....	592
Examples.....	592
Moving caret to the beginning of paragraph.....	592
OnDropFile Example.....	593
TDBRichView.....	594
Properties.....	596
TDBRichView.AllowMarkdown.....	598
TDBRichView.AutoDeleteUnusedStyles.....	599
TDBRichView.AutoDisplay.....	599
TDBRichView.CodePage.....	599
TDBRichView.DataField.....	600
TDBRichView.DataSource.....	600
TDBRichView.Field.....	600
Methods.....	600
TDBRichView.Create.....	604
TDBRichView.Destroy.....	604
TDBRichView.LoadField.....	604
Events.....	605
TDBRichViewEdit.....	606
Properties.....	608
TDBRichViewEdit.AllowMarkdown.....	611
TDBRichViewEdit.AutoDeleteUnusedStyles.....	611
TDBRichViewEdit.AutoDisplay.....	612
TDBRichViewEdit.CodePage.....	612
TDBRichViewEdit.DataField.....	613
TDBRichViewEdit.DataSource.....	613
TDBRichViewEdit.Field.....	613
TDBRichViewEdit.FieldFormat.....	613
TDBRichViewEdit.IgnoreEscape.....	614
TDBRichViewEdit.ReadOnly.....	614
Methods.....	614
TDBRichViewEdit.Change.....	621
TDBRichViewEdit.Create.....	621
TDBRichViewEdit.Destroy.....	621
TDBRichViewEdit.LoadField.....	622
Events.....	622
Examples.....	624
Using viewer-style methods in DBRichViewEdit.....	624
Inserting controls that can modify themselves in DBRichViewEdit.....	625
TRVPrintPreview.....	625
Properties.....	626
TRVPrintPreview.RVPrint.....	628
TRVPrintPreview.CachePageImage.....	628
Methods.....	628
Events.....	629
2. Additional Components.....	629
TRVStyle.....	630
Properties.....	633
TRVStyle.CheckpointColor, CheckpointColor.....	634

TRVStyle.Color	635
TRVStyle.CurrentItemColor	635
TRVStyle.DefCodePage	635
TRVStyle.DefTabWidth	636
TRVStyle.DefUnicodeStyle	636
TRVStyle.DisabledFontColor	637
TRVStyle.EndnoteNumbering	637
TRVStyle.FieldHighlightColor	637
TRVStyle.FieldHighlightType	638
TRVStyle.FloatingLineColor	638
TRVStyle.FontQuality	639
TRVStyle.FootnotePageReset	640
TRVStyle.FullRedraw	640
TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle	640
TRVStyle HoverColor	641
TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor, SelOpacity	641
TRVStyle.InvalidPicture	643
TRVStyle.JumpCursor	643
TRVStyle.LineSelectCursor	643
TRVStyle.LineWrapMode	644
TRVStyle.ListStyles	646
TRVStyle.LiveSpellingColor	646
TRVStyle.MainRVStyle	647
TRVStyle.PageBreakColor	647
TRVStyle.ParaStyles	648
TRVStyle.SelectionHandleKind	648
TRVStyle.SelectionMode	649
TRVStyle.SelectionStyle	649
TRVStyle.SoftPageBreakColor	651
TRVStyle.SpacesInTab	652
TRVStyle.SpecialCharactersColor	652
TRVStyle.StyleTemplates	652
TRVStyle.TextBackgroundKind	653
TRVStyle.TextEngine	654
TRVStyle.TextStyles	654
TRVStyle.Units	655
TRVStyle.UnitsPixelsPerInch	655
TRVStyle.UseSound	656
Methods	656
TRVStyle.AddTextStyle	657
TRVStyle.Create	657
TRVStyle.ConvertTo*	657
TRVStyle.DeleteTextStyle	658
TRVStyle.Destroy	658
TRVStyle.GetAs*	658
TRVStyle.*ToUnits	659
TRVStyle.GetListStyleClass	660
TRVStyle.GetParaStyleClass	661
TRVStyle.GetTextStyleClass	661
TRVStyle.LoadINI	661
TRVStyle.FindTextStyle, FindParaStyle	661
TRVStyle.LoadReg	662
TRVStyle.SaveCSS	662
TRVStyle.SaveCSSToStream	663
TRVStyle.SaveINI	663
TRVStyle.SaveReg	664
Events	664
TRVStyle.OnAfterApplyStyle	664

TRVStyle.OnApplyStyle	665
TRVStyle.OnApplyStyleColor	666
TRVStyle.OnDrawCheckpoint	667
TRVStyle.OnDrawPageBreak	668
TRVStyle.OnDrawParaBack	670
TRVStyle.OnDrawStyleText	671
TRVStyle.OnDrawTextBack	673
TRVStyle.OnStyleHoverSensitive	674
Classes of properties - Style collections	675
TFontInfos	675
Properties	676
TFontInfos.InvalidItem	676
TFontInfos.Items	677
TFontInfos.PixelsPerInch	677
Methods	677
TFontInfos.Add	678
TFontInfos.AddFont	678
TFontInfos.AddFontEx	678
TFontInfos.AssignTo	678
TFontInfos.Create	678
TFontInfos.FindStyleWithCharset	679
TFontInfos.FindStyleWithColor	679
TFontInfos.FindStyleWithFont	679
TFontInfos.FindStyleWithFontName	680
TFontInfos.FindStyleWithFontSize	680
TFontInfos.FindStyleWithFontStyle	681
TFontInfos.FindSuchStyle, FindSuchStyleEx	681
TParaInfos	682
Properties	682
TParaInfos.InvalidItem	683
TParaInfos.Items	683
Methods	683
TParaInfos.Add	684
TParaInfos.AssignTo	684
TParaInfos.Create	684
TParaInfos.FindStyleWithAlignment	684
TParaInfos.FindSuchStyle, FindSuchStyleEx	685
TRVListInfos	685
Properties	686
TRVListInfos.Items	686
Methods	686
TRVListInfos.FindSuchStyle	686
TRVListInfos.FindStyleWithLevels	687
TRVListInfos.AssignTo	687
TRVListInfos.Add	687
TRVStyleTemplateCollection	688
Properties	688
TRVStyleTemplateCollection.DefStyleName	688
TRVStyleTemplateCollection.ForbiddenIds	689
TRVStyleTemplateCollection.Items	689
TRVStyleTemplateCollection.NormalStyleTemplate	689
Methods	689
TRVStyleTemplateCollection.AssignStyleTemplates	690
TRVStyleTemplateCollection.AssignToStrings	690
TRVStyleTemplateCollection.ClearTextFormat, ClearParaFormat	690
TRVStyleTemplateCollection.FindById, FindItemById	691
TRVStyleTemplateCollection.FindByName, FindItemByName	691
TRVStyleTemplateCollection.InsertFromRVST	691

TRVStyleTemplateCollection.LoadFromRVST, SaveToRVST	692
TRVStyleTemplateCollection.LoadFromStreamRVST, SaveToStreamRVST.....	692
TRVStyleTemplateCollection.ResetNameCounter	693
Classes of properties - Styles.....	693
TCustomRVInfo, TCustomRVTextOrParaInfo.....	693
Properties	694
TCustomRVInfo.Standard.....	695
TCustomRVInfo.StyleName.....	696
TCustomRVFontOrParaInfo.StyleTemplateId.....	696
TCustomRVFontInfo	696
Properties	698
TCustomRVFontInfo.BackColor.....	698
TCustomRVFontInfo.BiDiMode.....	699
TCustomRVFontInfo.CharScale.....	699
TCustomRVFontInfo.Charset.....	700
TCustomRVFontInfo.CharSpacing.....	700
TCustomRVFontInfo.Color	701
TCustomRVFontInfo.EmptyWidth.....	701
TCustomRVFontInfo.FontName.....	701
TCustomRVFontInfo HoverBackColor.....	702
TCustomRVFontInfo HoverColor.....	702
TCustomRVFontInfo HoverEffects.....	703
TCustomRVFontInfo HoverUnderlineColor.....	703
TCustomRVFontInfo.JumpCursor.....	704
TCustomRVFontInfo.Options.....	704
TCustomRVFontInfo.ParaStyleTemplateId.....	705
TCustomRVFontInfo.Protection	705
TCustomRVFontInfo.Size, SizeDouble.....	707
TCustomRVFontInfo.Style.....	708
TCustomRVFontInfo.StyleEx.....	708
TCustomRVFontInfo.SubSuperScriptType.....	708
TCustomRVFontInfo.UnderlineColor	709
TCustomRVFontInfo.UnderlineType.....	709
TCustomRVFontInfo.VShift.....	710
Methods	710
TCustomRVFontInfo.Assign	711
TCustomRVFontInfo.AssignTo.....	711
TCustomRVFontInfo.Create.....	711
TCustomRVFontInfo.IsEqual.....	711
TFontInfo	712
Properties	713
TFontInfo.Jump.....	714
TFontInfo.ModifiedProperties.....	714
TFontInfo.NextStyleNo.....	714
TFontInfo.Unicode.....	715
Methods	715
TFontInfo.Assign.....	715
TFontInfo.Create.....	716
TFontInfo.Draw.....	716
TFontInfo.IsEqual.....	717
TCustomRVParaInfo	718
Properties	719
TCustomRVParaInfo.Alignment, LastLineAlignment.....	719
TCustomRVParaInfo.Background.....	721
TCustomRVParaInfo.BiDiMode.....	721
TCustomRVParaInfo.Border	721
TCustomRVParaInfo.FirstIndent.....	722
TCustomRVParaInfo.LeftIndent.....	722

TCustomRVParaInfo.LineSpacing.....	722
TCustomRVParaInfo.LineSpacingType.....	723
TCustomRVParaInfo.Options.....	723
TCustomRVParaInfo.OutlineLevel.....	725
TCustomRVParaInfo.RightIndent.....	725
TCustomRVParaInfo.SpaceAfter.....	726
TCustomRVParaInfo.SpaceBefore.....	726
TCustomRVParaInfo.Tabs.....	726
Methods	726
TCustomRVParaInfo.Assign.....	726
TCustomRVParaInfo.Create.....	727
TCustomRVParaInfo.IsEqual.....	727
TParaInfo	727
Properties	728
TParaInfo.DefStyleNo.....	729
TParaInfo.ModifiedProperties.....	729
TParaInfo.NextParaNo.....	730
Methods	730
TParaInfo.Assign.....	730
TParaInfo.Create.....	731
TParaInfo.IsEqual.....	731
TRVListInfo	731
Properties	732
TRVListInfo.Levels.....	732
TRVListInfo.OneLevelPreview.....	732
Methods	732
TRVListInfo.AllNumbered.....	733
TRVListInfo.HasNumbering.....	733
TRVStyleTemplate	733
Properties	734
TRVStyleTemplate.Id.....	734
TRVStyleTemplate.Kind.....	735
TRVStyleTemplate.Name.....	735
TRVStyleTemplate.NextId, Next.....	736
TRVStyleTemplate.ParaStyle.....	737
TRVStyleTemplate.ParentId, Parent.....	737
TRVStyleTemplate.QuickAccess.....	737
TRVStyleTemplate.TextStyle.....	737
TRVStyleTemplate.ValidParaProperties.....	738
TRVStyleTemplate.ValidTextProperties.....	738
Methods	738
TRVStyleTemplate.ApplyToParaStyle.....	738
TRVStyleTemplate.ApplyToTextStyle.....	739
TRVStyleTemplate.UpdateModifiedTextStyleProperties, UpdateModifiedParaStyleProperties.....	740
Classes of properties of a paragraph style.....	741
TRVBackgroundRect	741
Properties	742
TRVBackgroundRect.BorderOffsets.....	742
TRVBackgroundRect.Color.....	742
TRVBackgroundRect.Opacity.....	742
TRVBorder	742
Properties	743
TRVBorder.BorderOffsets.....	743
TRVBorder.Color.....	744
TRVBorder.InternalWidth.....	744
TRVBorder.Style.....	744
TRVBorder.VisibleBorders.....	744
TRVBorder.Width.....	744

TRVTabInfos	745
Properties	745
TRVTabInfos.Items	745
Methods	745
TRVTabInfos.AddFrom.....	746
TRVTabInfos.DeleteList.....	746
TRVTabInfos.Find.....	746
TRVTabInfos.Intersect.....	746
TRVTabInfo	746
Properties	747
TRVTabInfo.Align.....	747
TRVTabInfo.Leader.....	747
TRVTabInfo.Position.....	748
Classes of properties of a list style.....	748
TRVListLevelCollection	749
Properties	749
TRVListLevelCollection.Items	749
Methods	749
TRVListLevelCollection.Add	749
TRVListLevel	750
Properties	750
TRVListLevel.FirstIndent.....	751
TRVListLevel.Font.....	751
TRVListLevel.FormatString.....	752
TRVListLevel.FormatStringW.....	753
TRVListLevel.ImageIndex.....	753
TRVListLevel.ImageList.....	754
TRVListLevel.ImageWidth, ImageHeight.....	754
TRVListLevel.LeftIndent.....	755
TRVListLevel.ListType.....	755
TRVListLevel.MarkerAlignment.....	757
TRVListLevel.MarkerIndent.....	757
TRVListLevel.Options.....	758
TRVListLevel.Picture.....	759
TRVListLevel.StartFrom.....	759
Methods	759
TRVListLevel.HasNumbering.....	759
TRVPrint	759
Properties	764
TRVPrint.BestDPI	765
TRVPrint.ClipMargins	765
TRVPrint.FacingPages	765
TRVPrint.FixMarginsMode.....	766
TRVPrint.FooterY	766
TRVPrint.HeaderY	767
TRVPrint.Margins	768
TRVPrint.MirrorMargins.....	769
TRVPrint.Preview100PercentHeight.....	769
TRVPrint.Preview100PercentWidth.....	769
TRVPrint.TitlePage	769
TRVPrint.Units	770
TRVPrint.VirtualPrinter	770
Methods	770
TRVPrint.AssignDocParameters.....	771
TRVPrint.AssignSource	771
TRVPrint.ContinuousPrint.....	772
TRVPrint.ConvertToUnits	773
TRVPrint.Create	773

TRVPrint.DrawPage, DrawPageAt.....	773
TRVPrint.DrawPreview	773
TRVPrint.FormatPages	774
TRVPrint.GetFooterRect	775
TRVPrint.GetHeaderRect.....	775
TRVPrint.MakePreview	775
TRVPrint.MakeScaledPreview.....	776
TRVPrint.Print	776
TRVPrint.PrintPages	777
TRVPrint.SavePDF	778
TRVPrint.SetFooter	778
TRVPrint.SetHeader	779
Events	780
TRVPrint.OnFormatting	780
TRVPrint.OnPagePostpaint.....	781
TRVPrint.OnPagePrepaint.....	782
TRVPrint.OnSendingToPrinter.....	782
Classes of properties.....	783
TRVirtualPrinterProperties.....	783
Properties	783
TRVirtualPrinterProperties.Active.....	784
TRVirtualPrinterProperties.PixelsPerInch.....	784
TRVirtualPrinterProperties.PageWidth, PageHeight.....	784
TRVSpellChecker	784
Properties	786
TRVSpellChecker.AvailableLanguages.....	786
TRVSpellChecker.DialogShown.....	787
TRVSpellChecker.Language.....	787
TRVSpellChecker.LanguageIndex.....	787
TRVSpellChecker.SpellFormStyle.....	788
Methods	788
TRVSpellChecker.AddToAutoreplaceList.....	788
TRVSpellChecker.AddToDictionary.....	789
TRVSpellChecker.AddToIgnoreList.....	789
TRVSpellChecker.CurrentLanguage.....	789
TRVSpellChecker.Execute.....	789
TRVSpellChecker.FillLanguageNames.....	790
TRVSpellChecker.GetAutoCorrection.....	790
TRVSpellChecker.GetLanguageNameFromCode.....	790
TRVSpellChecker.GuessCompletions.....	791
TRVSpellChecker.IsAvailable.....	791
TRVSpellChecker.IsWordValid.....	791
TRVSpellChecker.RegisterAsService, UnregisterAsService.....	791
TRVSpellChecker.RegisterEditor, UnregisterEditor	792
TRVSpellChecker.StartLiveSpelling.....	792
TRVSpellChecker.Suggest.....	792
TRVSpellChecker.SupportsFeatures.....	793
Events	793
TRVSpellChecker.OnSpellForm.....	793
TRVSpellChecker.OnSpellFormAction.....	794
TRVSpellChecker.OnSpellFormLocalize.....	795
TRVOfficeConverter	795
Properties	796
TRVOfficeConverter.ErrorCode.....	796
TRVOfficeConverter.ExcludeDocX-Converter	797
TRVOfficeConverter.ExcludeHTML-Converter.....	797
TRVOfficeConverter.ExportConverters	798
TRVOfficeConverter.ExtensionsInFilter.....	798

TRVOfficeConverter.ImportConverters.....	798
TRVOfficeConverter.PreviewMode.....	799
TRVOfficeConverter.Stream.....	799
Methods	799
TRVOfficeConverter.Create.....	799
TRVOfficeConverter.Destroy.....	799
TRVOfficeConverter.ExportRTF.....	800
TRVOfficeConverter.ExportRV.....	800
TRVOfficeConverter.GetExportFilter	801
TRVOfficeConverter.GetImportFilter	801
TRVOfficeConverter.ImportRTF.....	801
TRVOfficeConverter.ImportRV.....	802
TRVOfficeConverter.IsValidImporter.....	803
Events	803
TRVOfficeConverter.OnConverting.....	803
Classes of properties.....	803
TRVOfficeCnvList	803
TRVOfficeConverterInfo.....	804
TRVReportHelper	804
Properties	805
TRVReportHelper.AllowTransparentDrawing.....	806
TRVReportHelper.TargetPixelsPerInch.....	806
TRVReportHelper.RichView.....	807
Methods	807
TRVReportHelper.DrawPage.....	808
TRVReportHelper.DrawPageAt.....	808
TRVReportHelper.Finished.....	809
TRVReportHelper.FormatNextPage.....	809
TRVReportHelper.GetLastPageHeight.....	809
TRVReportHelper.Init	809
TRVReportHelper.SavePDF.....	810
TRVReportHelper.UpdatePageNumbers	810
Events	811
TRVReportHelper.OnGetPageNumber.....	811
TRVDataSourceLink	811
Properties	812
TRVDataSourceLink.DataSource.....	812
TRVDataSourceLink.Editor.....	812
Methods	813
TRVDataSourceLink.Execute.....	813
3-Components ancestor classes	813
TRVScroller	813
Properties	814
TRVScroller.BorderStyle.....	816
TRVScroller.HScrollMax.....	816
TRVScroller.HScrollPos	816
TRVScroller.HScrollVisible.....	816
TRVScroller.InplaceEditor	817
TRVScroller.NoVScroll	817
TRVScroller.Tracking	818
TRVScroller.UseXPThemes.....	818
TRVScroller.VScrollMax.....	818
TRVScroller.VScrollPos	819
TRVScroller.VScrollVisible.....	819
TRVScroller.WheelStep	819
Methods	820
TRVScroller.Create	820

TRVScroller.ScrollTo[Position].....	820
Events	820
TRVScroller.OnHScrolled.....	821
TRVScroller.OnVScrolled.....	821
TCustomRVPrint	822
Properties	822
TCustomRVPrint.ColorMode.....	822
TCustomRVPrint.DarkMode.....	823
TCustomRVPrint.EndAt	823
TCustomRVPrint.FirstPageNo.....	823
TCustomRVPrint.IgnorePageBreaks.....	823
TCustomRVPrint.LastPageNo.....	824
TCustomRVPrint.MaxPrintedItemNo.....	824
TCustomRVPrint.MetafileCompatibility.....	824
TCustomRVPrint.MetafilePixelsPerInch.....	825
TCustomRVPrint.MinPrintedItemNo.....	825
TCustomRVPrint.NoMetafiles	825
TCustomRVPrint.PageNoFromNumber	826
TCustomRVPrint.PagesCount.....	826
TCustomRVPrint.PreviewCorrection.....	826
TCustomRVPrint.StartAt.....	826
TCustomRVPrint.TransparentBackground.....	827
Methods	827
TCustomRVPrint.CanSavePDF.....	827
TCustomRVPrint.Clear	827
TCustomRVPrint.GetFirstItemOnPage.....	827
TCustomRVPrint.GetPageNo.....	828
TCustomRVPrint.UpdatePalettInfo.....	828
Events	829
TCustomRVPrint.OnAfterPrintImage.....	829
TCustomRVPrint.OnDrawCheckpoint.....	830
TCustomRVPrint.OnDrawHyperlink.....	830
TCustomRVPrint.OnPrintComponent.....	831
TCustomRVPrintPreview	832
Properties	834
TCustomRVPrintPreview.ClickMode.....	836
TCustomRVPrintPreview.Color, Fill	836
TCustomRVPrintPreview.DarkModeUI	837
TCustomRVPrintPreview.MarginsPen, MarginsStroke.....	837
TCustomRVPrintPreview.PageBorderColor	837
TCustomRVPrintPreview.PageBorderWidth.....	838
TCustomRVPrintPreview.PageNo.....	838
TCustomRVPrintPreview.PrintableAreaPen, PrintableAreaStroke.....	838
TCustomRVPrintPreview.ShadowColor.....	839
TCustomRVPrintPreview.ShadowWidth.....	839
TCustomRVPrintPreview.ZoomInCursor	839
TCustomRVPrintPreview.ZoomMode.....	840
TCustomRVPrintPreview.ZoomOutCursor	840
TCustomRVPrintPreview.ZoomPercent.....	841
Methods	841
TCustomRVPrintPreview.Create.....	841
TCustomRVPrintPreview.First.....	841
TCustomRVPrintPreview.Last.....	842
TCustomRVPrintPreview.Next.....	842
TCustomRVPrintPreview.Prev.....	842
TCustomRVPrintPreview.SetZoom.....	843
Events	843
TCustomRVPrintPreview.OnZoomChanged.....	843

Part VII Item types

844

1. Table - TRVTableItemInfo	846
Properties	848
TRVTableItemInfo.BackgroundImage.....	849
TRVTableItemInfo.BackgroundImageFileName.....	850
TRVTableItemInfo.BackgroundStyle.....	850
TRVTableItemInfo.BestWidth.....	850
TRVTableItemInfo.BorderColor.....	851
TRVTableItemInfo.BorderHSpacing.....	851
TRVTableItemInfo.BorderLightColor.....	852
TRVTableItemInfo.BorderStyle.....	852
TRVTableItemInfo.BorderVSpacing.....	853
TRVTableItemInfo.BorderWidth.....	853
TRVTableItemInfo.CellBorderColor.....	853
TRVTableItemInfo.CellBorderLightColor.....	854
TRVTableItemInfo.CellBorderStyle.....	854
TRVTableItemInfo.CellBorderWidth.....	855
TRVTableItemInfo.CellHPadding.....	855
TRVTableItemInfo.CellHSpacing.....	856
TRVTableItemInfo.CellOverrideColor.....	856
TRVTableItemInfo.CellPadding.....	856
TRVTableItemInfo.Cells.....	857
TRVTableItemInfo.CellVPadding.....	857
TRVTableItemInfo.CellVSpacing.....	857
TRVTableItemInfo.ColBandSize, RowBandSize.....	858
TRVTableItemInfo.ColCount.....	858
TRVTableItemInfo.Color.....	858
TRVTableItemInfo.HeadingRowCount.....	859
TRVTableItemInfo.HOutermostRule.....	859
TRVTableItemInfo.HRuleColor.....	860
TRVTableItemInfo.HRuleWidth.....	860
TRVTableItemInfo.Opacity.....	860
TRVTableItemInfo.Options.....	861
TRVTableItemInfo.PrintOptions.....	863
TRVTableItemInfo.RowCount.....	864
TRVTableItemInfo.Rows.....	864
TRVTableItemInfo.TextColSeparator.....	865
TRVTableItemInfo.TextRowSeparator.....	865
TRVTableItemInfo.VisibleBorders.....	865
TRVTableItemInfo.VOutermostRule.....	866
TRVTableItemInfo.VRuleColor.....	866
TRVTableItemInfo.VRuleWidth.....	867
TRVTableItemInfo's row and column colors.....	867
Methods	868
TRVTableItemInfo.AssignProperties.....	870
TRVTableItemInfo.CanMergeCells.....	870
TRVTableItemInfo.CanMergeSelectedCells.....	871
TRVTableItemInfo.Changed.....	871
TRVTableItemInfo.Create.....	872
TRVTableItemInfo.CreateEx.....	872
TRVTableItemInfo.DeleteCols.....	873
TRVTableItemInfo.DeleteEmptyCols.....	873
TRVTableItemInfo.DeleteEmptyRows.....	873
TRVTableItemInfo.DeleteRows.....	874
TRVTableItemInfo.DeleteSelectedCols.....	874
TRVTableItemInfo.DeleteSelectedRows.....	875

TRVTableItemInfo.Deselect.....	875
TRVTableItemInfo.Destroy.....	876
TRVTableItemInfo.EditCell.....	876
TRVTableItemInfo.GetCellAt.....	876
TRVTableItemInfo.GetEditedCell.....	877
TRVTableItemInfo.GetNormalizedSelectionBounds.....	877
TRVTableItemInfo.GetSelectionBounds.....	878
TRVTableItemInfo.InsertCols.....	878
TRVTableItemInfo.InsertColsLeft.....	879
TRVTableItemInfo.InsertColsRight.....	879
TRVTableItemInfo.InsertRows.....	880
TRVTableItemInfo.InsertRowsAbove.....	880
TRVTableItemInfo.InsertRowsBelow.....	880
TRVTableItemInfo.IsCellSelected.....	881
TRVTableItemInfo.LoadFromStream.....	881
TRVTableItemInfo.MergeCells.....	881
TRVTableItemInfo.MergeSelectedCells.....	882
TRVTableItemInfo.MoveRows.....	882
TRVTableItemInfo.SaveRowsToStream.....	883
TRVTableItemInfo.SaveToStream.....	883
TRVTableItemInfo.Select.....	884
TRVTableItemInfo.SelectCols.....	884
TRVTableItemInfo.SelectRows.....	884
TRVTableItemInfo.SetCellBackgroundImage.....	885
TRVTableItemInfo.SetCellBackgroundImageFileName.....	885
TRVTableItemInfo.SetCellBackgroundStyle.....	885
TRVTableItemInfo.SetCellBestHeight.....	886
TRVTableItemInfo.SetCellBestWidth.....	886
TRVTableItemInfo.SetCellBorderColor.....	886
TRVTableItemInfo.SetCellBorderLightColor.....	886
TRVTableItemInfo.SetCellColor.....	887
TRVTableItemInfo.SetCellHint.....	887
TRVTableItemInfo.SetCellOpacity.....	887
TRVTableItemInfo.SetCellOptions.....	887
TRVTableItemInfo.SetCellRotation.....	888
TRVTableItemInfo.SetCellTag.....	888
TRVTableItemInfo.SetCellVAlign.....	888
TRVTableItemInfo.SetCellVisibleBorders.....	889
TRVTableItemInfo.SetRow*.....	889
TRVTableItemInfo.SetTableVisibleBorders.....	889
TRVTableItemInfo.SplitSelectedCellsHorizontally.....	890
TRVTableItemInfo.SplitSelectedCellsVertically.....	890
TRVTableItemInfo.UnmergeCells.....	890
TRVTableItemInfo.UnmergeSelectedCells.....	891
Events.....	891
TRVTableItemInfo.OnCellEditing.....	892
TRVTableItemInfo.OnDrawBackground.....	893
TRVTableItemInfo.OnDrawBorder.....	893
Classes of properties.....	895
TRVTableCellData.....	895
Properties.....	896
TRVTableCellData.BackgroundImage.....	896
TRVTableCellData.BackgroundImageFileName.....	897
TRVTableCellData.BackgroundStyle.....	897
TRVTableCellData.BestHeight.....	898
TRVTableCellData.BestWidth.....	898
TRVTableCellData.BorderColor.....	899
TRVTableCellData.BorderLightColor.....	899

TRVTableCellData.Color.....	900
TRVTableCellData.ColSpan.....	900
TRVTableCellData.Hint.....	901
TRVTableCellData.IgnoreContentHeight.....	901
TRVTableCellData.Opacity.....	902
TRVTableCellData.Options.....	902
TRVTableCellData.Rotation.....	904
TRVTableCellData.RowSpan.....	905
TRVTableCellData.Tag.....	906
TRVTableCellData.VAlign.....	906
TRVTableCellData.VisibleBorders.....	906
Methods	907
TRVTableCellData.GetRVData	907
Additional properties of TRVTableCellData.....	908
TRVTableRow.....	909
Properties	909
TRVTableRow.Items.....	910
TRVTableRow.KeepTogether	910
TRVTableRow.PageBreakBefore.....	910
TRVTableRow.VAlign.....	910
Methods	911
TRVTableRows.....	911
Properties	911
TRVTableRows.Items.....	912
Methods	912
TRVTableRows.GetMainCell.....	912
2..Label - TRVLabelItemInfo	912
Properties	913
TRVLabelItemInfo.Alignment.....	914
TRVLabelItemInfo.Cursor	914
TRVLabelItemInfo.MinWidth.....	914
TRVLabelItemInfo.ProtectTextStyleNo.....	914
TRVLabelItemInfo.RemoveInternalLeading, RemoveSpaceBelow.....	915
TRVLabelItemInfo.Text.....	915
TRVLabelItemInfo.TextStyleNo.....	915
Methods	916
TRVLabelItemInfo.Create.....	916
TRVLabelItemInfo.CreateEx.....	916
3..Math expression - TRVMathItemInfo	917
Properties	918
TRVMathItemInfo.DisplayInline.....	918
TRVMathItemInfo.FontName.....	918
TRVMathItemInfo.FontSizeDouble.....	919
TRVMathItemInfo.Text.....	919
TRVMathItemInfo.TextColor	920
Methods	920
TRVMathItemInfo.Create.....	920
4..Numbered sequence - TRVSeqItemInfo	921
Properties	922
TRVSeqItemInfo.FormatString.....	923
TRVSeqItemInfo.NumberType.....	923
TRVSeqItemInfo.Reset.....	923
TRVSeqItemInfo.SeqName.....	924
TRVSeqItemInfo.StartFrom.....	924
Methods	924
TRVSeqItemInfo.Create.....	924
TRVSeqItemInfo.CreateEx.....	925

5..Ancestor for notes - TCustomRVNoteltemInfo	925
Properties	926
TCustomRVNoteltemInfo.Document.....	926
TCustomRVNoteltemInfo.NoteText.....	927
Methods	927
TCustomRVNoteltemInfo.CreateEx.....	927
TCustomRVNoteltemInfo.ReplaceDocumentEd	927
6..Endnote - TRVEndnoteltemInfo	928
Properties	929
Methods	929
TRVEndnoteltemInfo.Create.....	930
TRVEndnoteltemInfo.CreateEx.....	930
7..Footnote - TRVFootnoteltemInfo	930
Properties	932
Methods	932
TRVFootnoteltemInfo.Create.....	932
TRVFootnoteltemInfo.CreateEx.....	933
8..Sidenote - TRVSidenoteltemInfo	933
Properties	935
TRVSidenoteltemInfo.BoxProperties.....	935
TRVSidenoteltemInfo.BoxPosition.....	935
Methods	936
TRVSidenoteltemInfo.Create.....	936
TRVSidenoteltemInfo.CreateEx.....	936
Classes of properties	937
TRVBoxProperties.....	937
Properties	937
TRVBoxProperties.Background.....	938
TRVBoxProperties.Border.....	938
TRVBoxProperties.Height, HeightType.....	938
TRVBoxProperties.VAlign.....	939
TRVBoxProperties.Width, WidthType.....	939
TRVBoxPosition.....	940
Properties	941
TRVBoxPosition.HorizontalAnchor.....	941
TRVBoxPosition.HorizontalXXX.....	942
TRVBoxPosition.PositionInText.....	944
TRVBoxPosition.RelativeToCell.....	945
TRVBoxPosition.VerticalAnchor.....	945
TRVBoxPosition.VerticalXXX.....	946
9..Text box - TRVTextBoxItemInfo	947
Methods	948
TRVTextBoxItemInfo.Create.....	948
TRVTextBoxItemInfo.CreateEx.....	948
10..Reference to the parent note - TRVNoteReferenceltemInfo	949
Properties	950
Methods	951
TRVNoteReferenceltemInfo.Create.....	951
TRVNoteReferenceltemInfo.CreateEx.....	951
11..Ancestor for text fields - TCustomRVFieldItemInfo	952
Properties	952
TCustomRVPageNumberItemInfo.NumberType.....	952
12..Page Number - TRVPageNumberItemInfo	953
Methods	953
TRVPageNumberItemInfo.Create.....	953

TRVPageNumberItemInfo.CreateEx.....	954
13..Page Count - TRVPageCountItemInfo	954
Methods	955
TRVPageCountItemInfo.Create.....	955
TRVPageCountItemInfo.CreateEx.....	955
Part VIII Other classes	956
1..ERichViewError	957
2..TCustomRVData and others (TRichView documents)	957
TCustomRVData.Edit	959
TCustomRVData.GetRVData	960
TCustomRVData.GetSourceRVData	960
TCustomRVFormattedData.GetOriginEx	960
3..TPrintableRV	961
4..TRVBooleanRect	961
5..TRVCanvas	961
6..TRVControlsPainter	962
7..TRVDefaultLoadProperties	963
Properties	963
TRVDefaultLoadProperties.DefaultBulletFontName.....	963
TRVDefaultLoadProperties.DefaultFontName.....	964
TRVDefaultLoadProperties.DefaultFontSizeDouble.....	964
TRVDefaultLoadProperties.DefaultFontSizesDouble.....	964
TRVDefaultLoadProperties.DefaultMonoSpacedFontName.....	965
TRVDefaultLoadProperties.DefaultProperties	965
TRVDefaultLoadProperties.DefaultUnicodeSymbolFontName.....	965
TRVDefaultLoadProperties.GenericFontNames	965
TRVDefaultLoadProperties.ListMarkerIndentPix.....	968
8..TRVCustomMathEquationManager	968
Properties	968
TRVCustomMathEquationManager.SaveMathMLInHTML.....	969
TRVCustomMathEquationManager.SaveOMMLInDocX.....	969
9..TRVDeleteUnusedStylesData	969
10..TRVGraphic	970
11..TRVGraphicClass	971
12..TRVGraphicHandler	972
13..TRVIntegerList	974
14..TRVMathDocObject	974
15..TRVPicture	975
16..TRVRect	975
17..TRVReportHelperWithHeaderFooters	976
Properties	976
TRVReportHelperWithHeaderFooters.MainDoc.....	977
TRVReportHelperWithHeaderFooters.UseStyleTemplates.....	977
Methods	977
TRVReportHelperWithHeaderFooters.AssignToRVPrint.....	977
TRVReportHelperWithHeaderFooters.Create.....	977
TRVReportHelperWithHeaderFooters.GetFooter	978
TRVReportHelperWithHeaderFooters.GetHeader.....	978
TRVReportHelperWithHeaderFooters.Reset.....	978

18..TRVStringList	978
19..TRVThumbnailMaker	979
20..TRVUnitsRect	980

Part IX Procedures and functions 980

1..DrawControl and Others	981
2..Functions for TRVUnits conversion	981
3..Functions from RVGetText Unit	983
4..Functions from RVLinear Unit	984
5..Functions from RVMathItem Unit	986
6..GetRVESearchOptions	986
7..GetRVSearchOptions	987
8..HTML from RTF de-encapsulation	987
9..RV_GetPrinterDC	988
10..RVGetFirstEndnote and Others	989
11..RVGetNoteTextStyleNo	990
12..RVIsURL, RVIsEmail	991
13..RVMathEquationManager	991
14..RVOpenURL	991
15..Unicode Conversion	992

Part X Types 993

1..TCheckpointData	993
2..TRVAnsiString	993
3..TRVBiDiMode	993
4..TRVBorderStyle	994
5..TRVBreakStyle	995
6..TRVCellVAlign	996
7..TRVCodePage	996
8..TRVColor	996
9..TRVColorMode	997
10..TRVCoord, TRVCoordPoint, TRVCoordSize, TRVCoordRect	998
11..TRVDBFieldFormat	998
12..TRVDisplayOptions	999
13..TRVDocRotation	999
14..TRVEnumScope	999
15..TRVESearchOptions	999
16..TRVExtraltemProperty	1000
17..TRVExtraltemStrProperty	1005
18..TRVFloatPosition	1006
19..TRVFloatPositionKind	1006
20..TRVFloatProperty	1007
21..TRVFloatSize	1007

22..TRVFontCharset	1008
23..TRVFontInfoProperties	1008
24..TRVFontSize	1009
25..TRVReaderStyleMode	1009
26..TRVGraphicType	1010
27..TRVHType	1011
28..TRVHTMLEncoding	1011
29..TRVHTMLLength	1012
30..TRVHTMLSavingPart	1012
31..TRVIntProperty	1013
32..TRVItemBackgroundStyle	1013
33..TRVSpellFormAction	1014
34..TRVLength	1015
35..TRVLoadFormat	1015
36..TRVOleDropEffects	1015
37..TRVOpacity	1015
38..TRVPageNumberType	1016
39..TRVPageSet	1016
40..TRVParalInfoProperties	1017
41..TRVPDFMetadata	1018
42..TRVPenStyle	1018
43..TRVPixel96Length	1018
44..TRVPixelLength	1019
45..TRVPrintingStep	1019
46..TRVRawByteString	1019
47..TRVReaderStyleMode	1019
48..TRVSaveFormat	1020
49..TRVSaveOptions	1020
50..TRVScreenAndDevice, PRVScreenAndDevice	1024
51..TRVSearchOptions	1024
52..TRVSeqType	1026
53..TRVStrProperty	1026
54..TRVStyleLength	1027
55..TRVStyleTemplateId	1027
56..TRVStyleTemplateName	1027
57..TRVStyleUnits	1027
58..TRVTableBorderStyle	1029
59..TRVTag	1029
60..TRVTextDrawStates	1030
61..TRVUndoType	1031
62..TRVUnicodeString	1032
63..TRVUnits	1032

64..TRVAlign	1033
65..TRVYesNoAuto	1034
66..TRVZoomPercent	1035

Part XI Global constants and variables 1035

..1..Clipboard Related Constants and Variables	1037
..2..Color constants	1038
..3..Constants Related to HTML and Text	1040
..4..Constants Related to the Caret	1040
..5..DefaultRVFPixelsPerInch	1042
..6..Pen style constants	1042
..7..RichViewAllowCopyTableCells	1042
..8..RichViewAlternativePicPrint	1043
..9..RichViewApostropheInWord	1043
10..RichViewAutoAssignNormalStyleTemplate	1043
11..RichViewCompareStyleNames	1044
12..RichViewCSSUseSystemColors	1044
13..RichViewCtrlUpDownScroll	1044
14..RichViewDoNotCheckRVFStyleRefs	1045
15..RichViewDoNotMergeNumbering	1045
16..RichViewEditDefaultProportionalResize	1045
17..RichViewEditEnterAllowsEmptyMarkeredLines	1046
18..RichViewJustifyBeforeLineBreak	1046
19..RichViewMaxPictureCount	1046
20..RichViewMaxThumbnailCount	1047
21..RichViewPixelsPerInch	1047
22..RichViewResetStandardFlag	1047
23..RichViewSaveHyperlinksInDocXAsFields	1048
24..RichViewShowGhostSpaces	1048
25..RichViewStringFormatMethod	1048
26..RichViewTableAutoAddRow	1049
27..RichViewTableDefaultRTFAutofit	1049
28..RichViewUnicodeInput	1050
29..RV_CP_DEFAULT	1050
30..RVAlwaysShowPlaceholders	1051
31..RVControlsPainter	1051
32..RVDefaultLoadProperties	1051
33..rveipc*** constants	1051
34..RVEMPTYTAG	1056
35..rvespc*** constants	1056
36..RVGetSystemColor	1057
37..RVGraphicHandler	1057

38..RVIsCustomURL	1058
39..RVParagraphMarks	1058
40..rvs*** constants	1059
41..RVThumbnailMaker	1061
42..RVVisibleSpecialCharacters	1061
Part XII How to...	1062
..1..change page size and orientation	1063
..2..draw page numbers, headers and footers	1063
..3..implement commands like "make bold", "apply font", etc.	1063
..4..implement Find and Replace dialogs	1064
..5..switch insert/overtyp e mode	1064
..6..move the caret to the beginning or to the end of document in editor	1065
..7..make Unicode editor	1066
..8..make a plain text editor	1066
..9..implement smart indenting	1068
10..combine several TRichView documents in one	1068
11..use third-party graphic classes with TRichView	1068
12..create a chat window	1071
13..remove text formatting	1071
Index	1074

1 TRichView reference

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

Version 23 (what's new? ³⁴)

VCL/FireMonkey ¹⁵⁵ /LCL ¹⁵⁴ Components

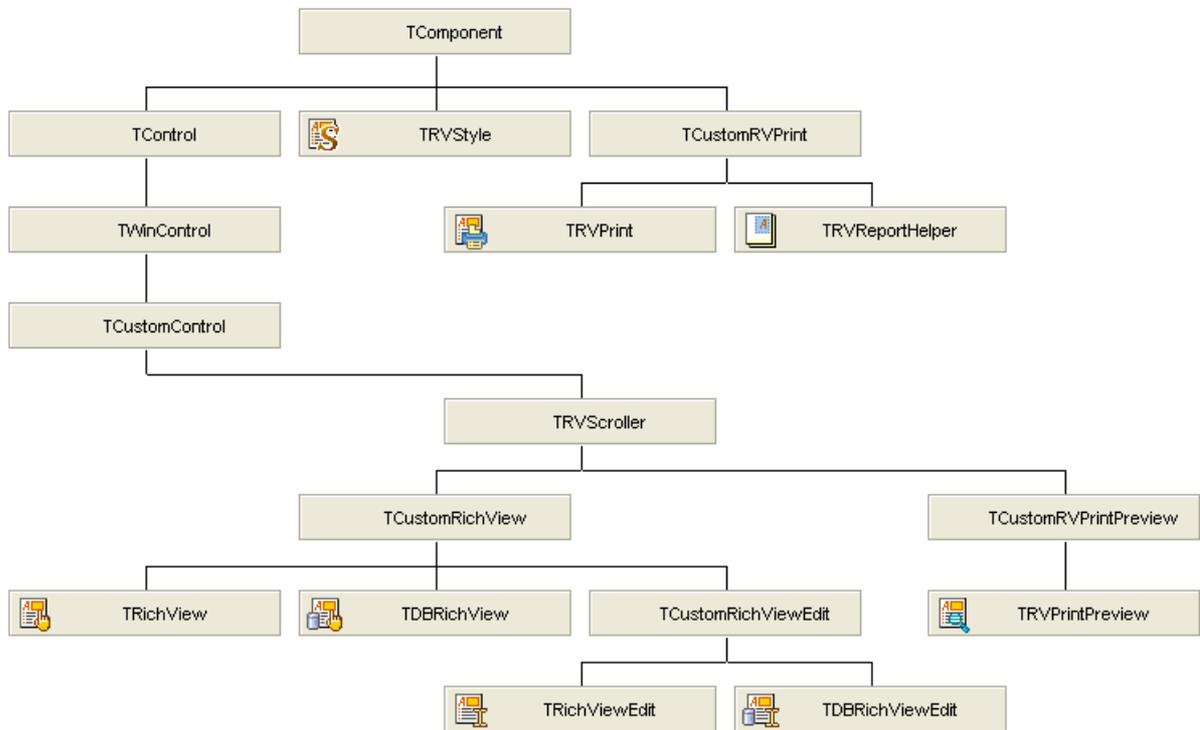
Main Components

Icon	Class Name	Description
	TRVStyle ⁶³⁰	Component for defining visual appearance of TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶ controls. It contains some properties which can be used by several TRichView controls.
	TRichView ²¹⁰	Component for displaying rich text documents
	TRichViewEdit ⁴⁶¹	Component for editing rich text documents
	TRVPrint ⁷⁵⁹	Component for printing documents contained in TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶
	TRVPrintPreview ⁶²⁵	Component allowing to show document as it will be printed by TRVPrint
	TRVReportHelper ⁸⁰⁴	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter ⁷⁹⁵	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL; obsolete]
	TRVSpellChecker ⁷⁸⁴	Component to check spelling in TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶ , TSRichViewEdit, TDBSRichViewEdit controls

Data-Aware Components [VCL and LCL]

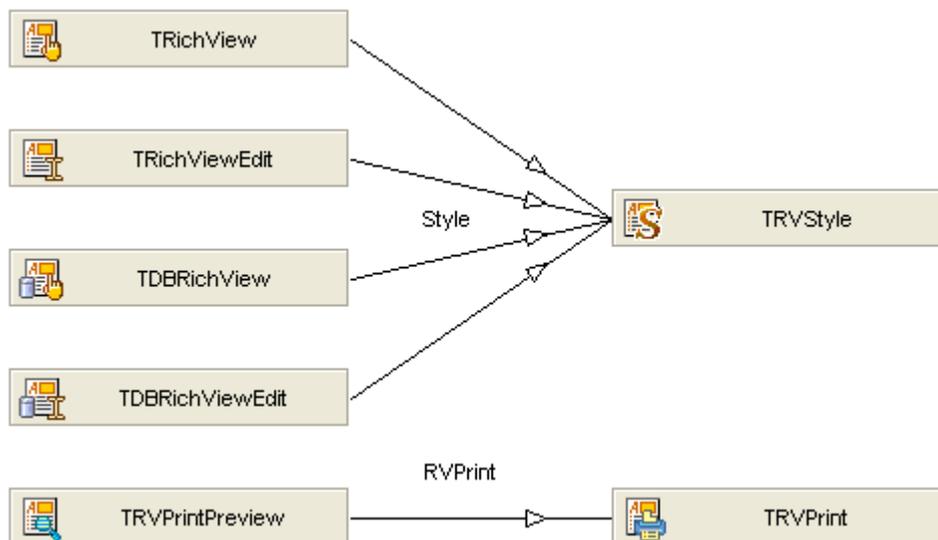
Icon	Class Name	Description
	TDBRichView ⁵⁹⁴	Data-aware version of TRichView [VCL and LCL]
	TDBRichViewEdit ⁶⁰⁶	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVDataSourceLink ⁸¹¹	Component allowing assigning TDataSource component to controls inserted in TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶ , 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 ³⁴
- overview of features ⁸⁴
- overview of item types in documents ¹⁵⁸

- overview of tables ¹⁸⁹
- how to... ¹⁰⁶²

2 Version history

Version History and Compatibility Issues

- New in version 23 ³⁴ (TRVSpellChecker, colors, Lazarus for Linux, equations in files)
- New in version 22 ³⁶ (FMX for Linux and iOS, decimal tabs, Skia4Delphi)
- New in version 21 ³⁸ (HTML import, reworked HTML export, LiveBindings, FMX for Android, update checker)
- New in version 20 ⁴⁰ (Markdown import, FMX for Windows and macOS)
- New in version 19 ⁴² (DocX import, Markdown export, TRVDataSourceLink)
- New in version 18 ⁴³ (Lazarus for Windows, all-Unicode, High DPI, EMU)
- New in version 17 ⁴⁶ (TRVMathItemInfo, row/column bands, header/footer)
- New in version 16 ⁴⁸ (Uniscribe, improved bi-di, vertical text, "distribute" alignment, semi-transparency)
- New in version 15 ⁵¹ (DocX export, text boxes and sidenotes, touch screen)
- New in version 14 ⁵³ (StyleTemplates, cell rotation, 64-bit, VCL styles)
- New in version 13 ⁵⁷ (new line wrapping, hidden text, twips)
- New in version 12 ⁵⁹ (left- and right-aligned objects, headings, showing more special characters)
- New in version 11 ⁶⁰ (Unicode)
- New in version 10 ⁶³ (animations, smart pop-ups)
- New in version 1.9 ⁶⁷ (tabs, live spelling check, showing special characters)
- New in version 1.8 ⁷⁰ (drag and drop, table background, mouse resizing)
- New in version 1.7 ⁷² (bullets and numbering)
- New in version 1.6 ⁷⁵ (TRVOfficeConverter, TRVReportHelper)
- New in version 1.5 ⁷⁷ (RTF import, storing styles in RVF, hypertext pictures)
- New in version 1.4 ⁸⁰ (tables)
- New in version 1.3 ⁸¹ (undo/redo, Unicode)
- New in version 1.2 ⁸² (paragraph borders and background, page breaks, search upwards)

2.1 New in version 23

This topic included changes introduced in v23.1

Compatibility issues

- New parameter (DarkMode) in TRVStyle.OnDrawStyleText ⁶⁷¹. New parameters (Printing, ColorMode, DarkMode) in TRVStyle.OnApplyStyleColor ⁶⁶⁶. If you use these events, you need to correct declarations of event handlers accordingly.
- New parameter (DarkMode) in TFontInfo.Draw ⁷¹⁶.
- If the commercial version of Adit Math Engine is used, equation objects ¹⁸⁷ 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⁽⁷⁸⁴⁾. 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⁽¹⁵⁴⁾ (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`⁽²²⁸⁾ inverts luminance of colors
- `UseFMXThemes`⁽²⁵⁶⁾ [FMX] allows using text color from FireMonkey style.

New property of TRVPrint and TRVReportHelper:

- `DarkMode`⁽⁸²³⁾ inverts luminance of colors (when drawing on a canvas or printing)

New property of TRVPrintPreview:

- `DarkModeUI`⁽⁸³⁷⁾ inverts luminance of colors in user interface of this component.

New global functional variable: `RVGetSystemColor`⁽¹⁰⁵⁷⁾ 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 ⁽¹⁸⁷⁾ 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 returned by RVMathEquationManager ⁽⁹⁹¹⁾ function.

Files

Markdown

New optional parameter CodePage for TRichView.SaveMarkdown ⁽³⁴⁰⁾, SaveMarkdownToStream ⁽³⁴¹⁾ allows to specify text encoding.

Universal loading methods

New optional CodePage parameter of LoadFromFile, LoadFromFileEx ⁽³²⁰⁾ and LoadFromStream, LoadFromStreamEx ⁽³²¹⁾ specifies encoding for loading plain text and Markdown.

Database and LiveBindings

New option is added to TRVDBFieldFormat ⁽⁹⁹⁸⁾: *rvdbMarkdown*. It allows saving data to a database field in Markdown format.

New properties: TDBRichView ⁽⁵⁹⁴⁾.AllowMarkdown ⁽⁵⁹⁸⁾, TDBRichViewEdit ⁽⁶⁰⁶⁾.AllowMarkdown ⁽⁶¹¹⁾, TRichView ⁽²¹⁰⁾.Document ⁽²³²⁾.AllowMarkdown ⁽⁴²²⁾ allow loading Markdown from database fields (instead of plain text)

New properties: TDBRichView ⁽⁵⁹⁴⁾.CodePage ⁽⁵⁹⁹⁾, TDBRichViewEdit ⁽⁶⁰⁶⁾.CodePage ⁽⁶¹²⁾, TRichView ⁽²¹⁰⁾.Document ⁽²³²⁾.CodePage ⁽⁴²³⁾ specify codepage for saving/loading Markdown and plain text from database fields.

2.2 New in version 22

Compatibility issues

- In FireMonkey, TRVGraphicFM (also accessible as TRVGraphic ⁽⁹⁷⁰⁾) is an abstract class now. Actual classes representing images are inherited from it (such as TRVRasterGraphicFM, also accessible as TRVBitmap)
- TRVGraphicClass ⁽⁹⁷¹⁾ declaration is changed for FireMonkey

RAD Studio 12 Athens

Delphi and C++Builder 12 are supported.

Skia

TRichView supports Skia4Delphi⁽¹⁵⁷⁾. 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 (TSvgGraphic is registered as SVG class for RVGraphicHandler⁽¹⁰⁵⁷⁾)

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⁽³⁸⁶⁾ 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⁽²⁴⁶⁾.ReadNoteNumberingType⁽⁴⁵⁷⁾ = *True* to enable.

HTML

- Saving: new option for TRichView.HTMLSaveProperties⁽²³⁷⁾.ImageOptions⁽⁴³⁴⁾: *rvhtmlsiolmagesizeattributes*.
- Loading: new property TRichView.HTMLReadProperties⁽²³⁷⁾.XHTMLCheck⁽⁴²⁸⁾ turns XHTML stricter syntax checking on/off.

PDF

- [FireMonkey with Skia4Delphi⁽¹⁵⁷⁾] New methods: TRVPrint.SavePDF⁽⁷⁷⁸⁾ and TRVReportHelper.SavePDF⁽⁸¹⁰⁾.

Graphics

- SVG images are supported using Skia4Delphi. FireMonkey: TRVSvgImageSkiaFM⁽⁹⁷⁰⁾ class (include fmxRVSkiaFM unit in your project); VCL: TSvgGraphic (include RVSkia unit in your project).
- New formats are added in TRVGraphicType⁽¹⁰¹⁰⁾: JPEG XR and WebP. RVGraphicHandler⁽¹⁰⁵⁷⁾.LoadFromStream and GetDataGraphicType⁽⁹⁷²⁾ can detect these formats.
- [FireMonkey] RVGraphicHandler⁽¹⁰⁵⁷⁾.GetFileDialogFilter⁽⁹⁷²⁾ returns a file filter string that includes both bitmap and non-bitmap (such as SVG) formats that can be loaded.

Other

Change: `Table.DeleteEmptyCols`⁽⁸⁷³⁾ and `DeleteEmptyRows`⁽⁸⁷³⁾ keep selection in the table.

New tab alignment⁽⁷⁴⁷⁾: decimal align to align floating point numbers in columns.

New property for images: `rvepSmoothScaling`⁽¹⁰⁰⁰⁾, it allows to turn smooth scaling on/off.

2.3 New in version 21

This topic included changes introduced in v21.1.

Compatibility issues

`TRichViewEdit.SetItemTextEd`⁽⁵⁶⁴⁾ (and related methods) calls `OnItemTextEd`⁽⁵⁷⁸⁾ event.

`TRVStyle.SaveCSS`⁽⁶⁶²⁾ and `SaveCSSToStream`⁽⁶⁶³⁾ parameters changed (`rvcssUTF8` is removed from `AOptions`, new `Encoding` parameter is added).

`TRichView.SavePicture`⁽³⁴²⁾ 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`⁽³³⁷⁾, `SaveHTMLToStreamEx`⁽³⁴⁰⁾, old versions of `SaveHTML`⁽³³⁶⁾, `SaveHTMLToStream`⁽³⁴⁰⁾ 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`⁽²³⁷⁾.

New methods:

- `TRichView.LoadHTML`⁽³²²⁾, `LoadHTMLFromStream`⁽³²⁴⁾;
- `TRichViewEdit.InsertHTMLFromFileEd`⁽⁵²¹⁾, `InsertHTMLFromStreamEd`⁽⁵²¹⁾;
- `TRichViewEdit.PasteHTML`⁽⁵³⁶⁾, `CanPasteHTML`⁽⁴⁹⁹⁾.

New event: `TRichView.OnImportFile`⁽³⁷⁸⁾.

`LoadFromFile`⁽³²⁰⁾, `LoadFromStream`⁽³²¹⁾, `Paste`⁽⁵³⁴⁾ methods support HTML.

`TDBRichView`⁽⁵⁹⁴⁾ and `TDBRichViewEdit`⁽⁶⁰⁶⁾ components support HTML.

HTML export

Old HTML saving methods are deprecated.

New property: `TRichView.HTMLSaveProperties`⁽²³⁷⁾.

New methods: `SaveHTML`⁽³³⁵⁾ and `SaveHTMLToStream`⁽³³⁸⁾. They use properties from `HTMLSaveProperties`⁽²³⁷⁾ and `DocParameters`⁽²³⁰⁾.

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`⁽²³⁹⁾.`SaveOptions`⁽⁴⁴⁸⁾: `rvmdsInlineImages`.

New methods for loading and saving

`LoadFromFile/LoadFromFileEx`⁽³²⁰⁾ 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`⁽²³²⁾. This property can be linked to a DB field using LiveBindings. It contains sub-properties defining properties of this link.

The events `OnNewDocument`⁽³⁸⁴⁾, `OnLoadDocument`⁽³⁸⁴⁾, `OnLoadCustomFormat`⁽³⁸³⁾, `OnSaveCustomFormat`⁽⁵⁸⁵⁾ 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⁽¹³²⁾ can be made both using third-party spelling checkers, and using a platform spelling check service (if available, see `CheckSpelling`⁽⁴⁷²⁾ property). Suggestions for corrections can be shown in a popup menu (automatically for a standard menu, or using `AddSpellingMenuItems`⁽⁴⁸⁸⁾ for a custom menu). Suggestions are requested either from a platform spelling check service, or using `OnGetSpellingSuggestions`⁽³⁷⁵⁾ event.

"SmartPopup" button can display not only a menu, but `TPopup` component (`Popup`⁽⁵⁹¹⁾ 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`⁽⁴⁷⁵⁾)

Tables

New `IgnoreContentHeight`⁽⁹⁰¹⁾ property of table cells.

Text search without selection

The new method `TRichView.StoreSearchResult`⁽³⁶⁷⁾ allows storing the position of the last found string from `OnTextFound`⁽⁴¹⁰⁾ event. This position can be used as a starting point for the next call of `TRichView.SearchText`⁽³⁴⁶⁾/`TRichViewEdit.SearchText`⁽⁵⁴³⁾, if `rvsroFromStored/rvseoFromStored` is included `SrchOptions` parameter.

Custom drawing

New event `TRichViewEdit.OnDrawCurrentLine`⁽⁵⁷⁴⁾ allows to highlight the current line.

Other

New optional parameters (`ImageWidth` and `ImageHeight`) in methods: `TRichView.AddPicture`⁽²⁷²⁾ and `AddHotPicture`⁽²⁶⁷⁾, `TRichViewEdit.InsertPicture`⁽⁵²⁶⁾ and `InsertHotPicture`⁽⁵¹⁹⁾.

2.4 New in version 20

Compatibility issues

To use equation objects⁽¹⁸⁷⁾ 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`⁽⁶³⁰⁾.`MainRVStyle`⁽⁶⁴⁷⁾ is assigned, not only `StyleTemplates`⁽⁶⁵²⁾, 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`⁽²¹⁰⁾, `TRichViewEdit`⁽⁴⁶¹⁾, `TRVStyle`⁽⁶³⁰⁾, `TRVPrint`⁽⁷⁵⁹⁾, `TRVReportHelper`⁽⁸⁰⁴⁾ 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⁽¹⁵⁵⁾.

Markdown

New methods for loading **Markdown** in `TRichView`: `LoadMarkdown`⁽³²⁵⁾, `LoadMarkdownFromStream`⁽³²⁵⁾, and inserting in `TRichViewEdit`: `InsertMarkdownFromFileEd`⁽⁵²³⁾, `InsertMarkdownFromStreamEd`⁽⁵²⁴⁾.

For `LoadFromStream`³²¹, 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`⁴³⁹ class.

RTF and DocX

New properties controlling RTF and DocX import:

- `CanReuseNumberedLists`⁴⁵²
- `ConvertSymbolFonts`⁴⁵³

Equation document objects

Both the free and the commercial version of Addit math engine are supported in `TRichView` equation objects¹⁸⁷.

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`³⁷⁰
- `TCustomRVPrint.OnAfterPrintImage`⁸²⁹

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`⁶⁴¹.

For `FireMonkey`, there is an `Option`²⁴⁰ to get selection color and opacity from the linked visual style: `rvoDefaultSelectionFill`.

Tables

New table operations:

- `TRVTableItemInfo.MoveRows`⁸⁸².

Other new features

- new list counter types: lower Greek and decimal-leading-zero for paragraph numbering⁷⁵⁵.
- new global variable `RichViewCtrlUpDownScroll`¹⁰⁴⁴, specifies how `Ctrl + Up` and `Ctrl + Down` keys are processed.
- new `TRichViewEdit.MoveCaret`⁵³⁴ method.
- HTML saving improvement: for paragraph numbering, values of auto-calculated counters are saved only if necessary

- HTML from RTF extraction ⁽⁹⁸⁷⁾

2.5 New in version 19

This topic includes changes introduced in version 19.1.

Compatibility issues

- In TDBRichViewEdit ⁽⁶⁰⁶⁾, unused styles are deleted not only if AutoDeleteUnusedStyles ⁽⁶¹¹⁾ = *True*, but also if FieldFormat ⁽⁶¹³⁾ = *rvdbRTF* or *rvdbDocX*.
- TRichView ⁽²¹⁰⁾.SavePicture ⁽³⁴²⁾ parameters are changed.
- TRichView.RTFReadProperties.ErrorCode is renamed to RTFErrorCode ⁽⁴⁵⁷⁾.
- Backspace in tables ⁽¹⁷³⁾ deletes the selected rows/columns (or the table itself, if all cells are selected). To turn off this feature, assign *False* to RichViewTableAutoAddRow ⁽¹⁰⁴⁹⁾.

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 ⁽³¹⁸⁾, LoadDocXFromStream ⁽³¹⁹⁾.

New methods of TRichViewEdit: InsertDocXFromStreamEd ⁽⁵¹⁸⁾, InsertDocXFromFileEd ⁽⁵¹⁸⁾.

TRichView.LoadFromStream ⁽³²¹⁾ can detect and load DocX.

TDBRichViewEdit ⁽⁶⁰⁶⁾ can store documents as DocX ⁽⁶¹³⁾.

Markdown

New methods for exporting TRichView documents to **Markdown** files or streams: SaveMarkdown ⁽³⁴⁰⁾, SaveMarkdownToStream ⁽³⁴¹⁾.

New property MarkdownProperties ⁽²³⁹⁾ contains sub-properties that specify Markdown saving options.

Other new features

New components:

 TRVDataSourceLink ⁽⁸¹¹⁾ helps to use TRichView or ScaleRichView components as containers for data-aware controls.

New event of TRichView component:

- OnWriteObjectProperties ⁽⁴¹³⁾ allows saving additional object properties in RTF, DocX, and HTML.
- OnTextFound ⁽⁴¹⁰⁾ allows a special processing of found text (instead of / in addition to selecting)
- New optional parameter in TRVReportHelper ⁽⁸⁰⁴⁾.DrawPageAt ⁽⁸⁰⁸⁾.

- New property `TRVReportHelper.TargetPixelsPerInch` allows overriding the target canvas's pixel depth.
- New optional parameter (`SmoothScroll`) in `TRVScroller.ScrollTo`.
- New option for text protection: `rvoFastDeleteProtectedText` option for `TRichViewEdit.EditorOptions`.
- High-quality image scaling is supported for transparent images too (for Delphi 2009+ and Lazarus)
- `TRVReportHelper` uses high-quality image scaling, if `AllowTransparentDrawing` and `Preview` parameter are `True`
- Backspace in tables deletes the selected rows/columns (or the table itself, if all cells are selected).

Changes

In `TRichViewEdit.OnItemResize` event, values of `Val1` and `Val2` parameters are changed.

`TRichViewEdit.SetCheckpointInfoEd` and `InsertCheckpoint` were changed from procedures to functions.

2.6 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` instead of `AddBreakEx`, `AddBreakTag`, `AddBreakExTag`.

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

Changed events:

- In `TRichView`: `OnAssignImageFileName`, `OnImportPicture`, `OnItemAction`, `OnItemHint`, `OnReadHyperlink`, `OnRVDbClick`, `OnRVRightClick`, `OnRVControlNeeded`, `OnRVFPictureNeeded`, `OnSaveComponentToFile`, `OnSaveDocXExtra`, `OnSaveHTMLExtra`, `OnSaveImage2`, `OnSaveItemToFile` (additionally, `Unicode: Boolean` parameter is removed), `OnSaveParaToHTML`, `OnSaveRTFExtra`, `OnSpellingCheck`, `OnWriteHyperlink`
- In `TRichViewEdit`: `OnDropFile`, `OnItemTextEdit`
- In `TRVStyle`: `OnDrawStyleText` (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` (`Baseline` and `PSaD`), `OnApplyStyle` (`PSaD`)
- In `TFontInfo`: `Draw` (`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⁽³⁹⁹⁾, 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⁽¹⁰⁴²⁾), however, the actual PPI is assigned to their Font.PixelsPerInch.

RichViewPixelsPerInch⁽¹⁰⁴⁷⁾ global variable and TRVStyle.TextStyles.PixelsPerInch⁽⁶⁷⁷⁾ are deprecated. Use TRichView.DocumentPixelsPerInch⁽²³³⁾ and TRVStyle.UnitsPixelsPerInch⁽⁶⁵⁵⁾.

TRVStyle.GetAsPixels⁽⁶⁵⁸⁾, TRVStyle.PixelsToUnits⁽⁶⁵⁹⁾, and functions for unit conversions⁽⁹⁸¹⁾ have additional parameters.

Sizes measured in pixels now use TRVStyle.UnitsPixelsPerInch⁽⁶⁵⁵⁾ instead of Screen.PixelsPerInch. This change affects some code, e.g. the example in RV_GetPrinterDC⁽⁹⁸⁸⁾;

Lazarus

TRichView is compatible with Lazarus⁽¹⁵⁴⁾ (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⁽⁷¹⁵⁾
- TRVStyle.DefUnicodeStyle⁽⁶³⁶⁾
- TRVRTFReaderProperties.UnicodeMode⁽⁴⁵⁹⁾

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⁽⁴⁷⁵⁾
- 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`¹⁰²⁷: EMU (English metric unit).

You can define DPI for all values measured in pixels in `TRVStyle.UnitsPixelsPerInch`⁶⁵⁵. 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`⁶⁵⁵, 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`²³³ property. This property allows to implement zooming.

Editing

A new (alternative) way to group editing operations for undo:

`BeginUndoGroup2..EndUndoGroup2`⁴⁹⁷ (or `BeginUndoCustomGroup2..EndUndoCustomGroup2`⁴⁹⁷). These methods allow grouping operations in several table cells.

New events

`TRichView.OnReadMergeField`³⁹⁰ 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`⁶²⁵ or in `ScaleRichView`). This change affected `TRVStyle.OnDrawStyleText`⁶⁷¹ event and `TFontInfo.Draw`⁷¹⁶ 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`²⁴⁷: *rvfolgnoreGraphicClasses*, 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 controls¹⁶⁸ from RVF PPI to the actual PPI; you can disable scaling by assigning 0 to the additional integer property¹⁰⁵¹ *rveipcDPIscalable* (but the actual PPI is still assigned to `Font.PixelsPerInch` of controls).

DocX

- `OnSaveDocXExtra`³⁹⁹, 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⁽¹¹⁹⁾. 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⁽⁹⁸³⁾ may return more than one characters, if they assemble a single glyph.
- some improvements in table layout algorithm.

2.7 New in version 17

This topic includes changes of v17.2.

Compatibility issues

- New parameter in TRVPageCountItemInfo⁽⁹⁵⁴⁾.CreateEx⁽⁹⁵⁵⁾
- 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⁽¹¹⁸⁾, the editor gives a higher priority to URL than it was before
- *rvddHTML* is included in the default value of TRichViewEdit⁽⁴⁶¹⁾.AcceptPasteFormats⁽⁴⁷⁰⁾ 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⁽²⁴⁶⁾.StoreImagesFileNames property is removed.
- Previously, the functions for enumerations of notes and text boxes⁽⁹⁸⁹⁾ returned hidden⁽¹⁰¹⁾ notes/boxes if *rvoShowHiddenText* was included in Options⁽²⁴⁰⁾, even if AllowHidden parameter was *False*. Now, AllowHidden parameter is processed strictly.
- TCustomRichView.OnURLNeeded event is removed (use OnWriteHyperlink⁽⁴¹¹⁾)
- TRichView.RTFReadProperties.SetHeader and SetFooter are deprecated. Use TRichView.SetHeader⁽³⁵⁶⁾ and SetFooter⁽³⁵⁵⁾ instead. Note: they affect not only RTF and DocX, but RVF as well.
- type of the Sender parameter in TRVReportHelper.OnDrawHyperlink⁽⁸³⁰⁾ and OnDrawCheckpoint⁽⁸³⁰⁾ is changed from TRVReportHelper to TCustomRVPrint.

Mathematical expressions⁽¹⁸⁷⁾

TRichView includes a new item type displaying mathematical formulas (TRVMathItemInfo⁽⁹¹⁷⁾). It requires Delphi XE4 or newer.

Tables⁽⁸⁴⁶⁾

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

- HeadingRowColor⁽⁸⁶⁷⁾, LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor;
- RowBandSize, ColBandSize⁽⁸⁵⁸⁾.

A new optional parameter (EditCellAfter) is added in DeleteRows⁽⁸⁷⁴⁾ and DeleteCols⁽⁸⁷³⁾ methods.
New property: CellOverrideColor⁽⁸⁵⁶⁾.
New event: OnDrawBackground⁽⁸⁹³⁾.

RVF⁽¹²⁴⁾

New DocObjects⁽²²⁹⁾ property allows saving custom objects in RVF.

Headers and footers

TRichView, SetHeader⁽³⁵⁶⁾ and SetFooter⁽³⁵⁵⁾ define documents as headers/footers. They can be used not only for RTF (as before), but also for RVF and HTML.
A new option⁽¹⁰²⁰⁾ is available for HTML saving, *rvsoHeaderFooter*.

Printing

New property TRVPrint⁽⁷⁵⁹⁾.VirtualPrinter⁽⁷⁷⁰⁾ allows drawing pages on any canvas using the specified page size and DPI. New methods DrawPage and DrawPageAt⁽⁷⁷³⁾.

New optional parameters in TRVPrint⁽⁷⁵⁹⁾.Print⁽⁷⁷⁶⁾ and PrintPages⁽⁷⁷⁷⁾ allow printing all/odd/even pages, in normal/reverse order.

"Widow and orphan control" is implemented, see *rvpaoWidowOrphanControl*⁽⁷²³⁾ option of paragraphs.

The following events and properties are moved from TRVReportHelper⁽⁸⁰⁴⁾ to its parent class: OnDrawCheckpoint⁽⁸³⁰⁾, OnDrawHyperlink⁽⁸³⁰⁾, NoMetafiles⁽⁸²⁵⁾. Now you can use them in TRVPrint⁽⁷⁵⁹⁾ as well.

A new property is added in TRVPrint⁽⁷⁵⁹⁾ and TRVReportHelper⁽⁸⁰⁴⁾: MetafilePixelsPerInch⁽⁸²⁵⁾.

TRVPageCountItemInfo⁽⁹⁵⁴⁾ now has NumberType⁽⁹⁵²⁾ property, like TRVPageNumberItemInfo⁽⁹⁵³⁾,

OLE Drag&drop⁽¹¹⁸⁾

New events of TRichViewEdit⁽⁴⁶¹⁾: OnBeforeOleDrop and OnAfterOleDrop⁽⁵⁷⁰⁾.

New option *rvDragDropPicturesFromLinks* for TRichViewEdit.EditorOptions⁽⁴⁷⁵⁾. 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⁽⁷⁰¹⁾. 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⁽¹⁰⁴⁶⁾ global variable.

Loading external images

A new option is added to `TRichView.Options`⁽²⁴⁰⁾: `rvoAssignImageFileNames`. If included, the component assigns "image file name" properties when reading external images. Previously, `RTFReadProperties.StoreImagesFileNames` was used for this purpose for external images in RTF; this property is removed.

New event: `OnAssignImageFileName`⁽³⁷¹⁾ allows modifying file names before assigning to items/cells.

Item properties

`rveipcPercentWidth` additional integer property⁽¹⁰⁵¹⁾ is added to controls⁽¹⁶⁸⁾.

`rvepSpacing`, `rvepBorderWidth`, `rvepBorderColor`, `rvepOuterHSpacing`, `rvepOuterVSpacing` integer properties⁽¹⁰⁰⁰⁾ work for labels⁽¹⁷⁶⁾.

`rveipcRemoveSpaceBelow` additional integer property⁽¹⁰⁵¹⁾ is added to labels⁽¹⁷⁶⁾.

Caret

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

New global variables controlling the caret size and position⁽¹⁰⁴⁰⁾ are added. The new options include:

- setting the caret width equal to the next character/item width;
- using `OnMeasureCustomCaret`⁽⁵⁷⁸⁾ event for a standard caret (not only for a custom caret);
- options for aligning wide carets horizontally.

Other changes

New `TRichView.UseVCLThemes`⁽²⁵⁷⁾ property allows using colors of active VCL theme (in Delphi XE2+)

New `TRichViewEdit` event: `OnDropFile`⁽⁵⁷⁶⁾. It's more easy to use then `OnDropFiles`⁽⁵⁷⁶⁾, and occurs not only on drag&drop, but also on pasting.

New `TRichView` event: `OnCMHintShow`⁽³⁷²⁾ allows displaying custom hints.

Text flow options (`ClearLeft`⁽²²⁵⁾ and `ClearRight`⁽²²⁶⁾) are shown as vertical lines around paragraph/line-break marks, if `rvoShowSpecialCharacters` is included in `TRichView.Options`⁽²⁴⁰⁾. Text flow options can be exported to DocX.

New method `TRVStyleTemplateCollection.InsertFromRVST`⁽⁶⁹¹⁾, it allows applying styles from a file to existing documents.

Packages are separated into runtime and designtime packages.

2.8 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`⁽⁷¹⁹⁾ = *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⁽⁶⁷¹⁾; you can return the old justification method by activating `RVOLDJUSTIFY` in `RV_Defs.inc`; however, this directive disables `ParaStyle.Alignment`⁽⁷¹⁹⁾ = *rvaDistribute* (it works like *rvaJustify*)
- `TRVStyle.OnDrawStyleText`⁽⁶⁷¹⁾ has new parameters.
- `RichViewSafeFormatting` constant is removed (replaced with `RichViewStringFormatMethod`⁽¹⁰⁴⁸⁾)
- 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`⁽⁶⁵⁴⁾ property.

By default, Uniscribe text output is not compatible with metafiles. You can turn on metafile compatibility for `TRVReportHelper`⁽⁸⁰⁴⁾ and `TRVPrint`⁽⁷⁵⁹⁾ components using `MetafileCompatibility`⁽⁸²⁴⁾ property.

New paragraph properties

New value for `Alignment`⁽⁷¹⁹⁾ – *rvaDistribute*. It aligns paragraph content to the both left and right sides by adding space between all characters.

New property `LastLineAlignment`⁽⁷¹⁹⁾ 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`⁽⁷⁰⁰⁾
- `ParaStyle.Alignment`⁽⁷¹⁹⁾ = *rvaJustify* (and *rvaDistribute*)
- `TRVPrint.PreviewCorrection`⁽⁸²⁶⁾
- 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 ⁽⁹⁰²⁾ are implemented for vertical text in table cells.

In addition to rotation ⁽⁹⁰⁴⁾ 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 ⁽²⁴⁷⁾: *rvfoSaveStyleTemplatesOnlyNames*. It allows creating multiple RVF documents using the same collection of StyleTemplates ⁽⁶⁵²⁾.

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 ⁽⁸⁰⁶⁾ allows using methods for semitransparent drawing. It is useful for drawing over other images.

See the topic about semitransparent objects ⁽¹²¹⁾.

Support of high DPI screen modes

Size of touch screen selection handles ⁽⁶⁴⁸⁾ 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⁽⁵⁴²⁾ deletes a page break from the current paragraph (not from the current item as before)

New RichViewSaveHyperlinksInDocXAsFields⁽¹⁰⁴⁸⁾ global variable.

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

2.9 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⁽⁹⁷²⁾)
 - (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⁽⁷⁷⁵⁾, GetFooterRect⁽⁷⁷⁵⁾.
- New values are added to TRVUndoType⁽¹⁰³¹⁾.

New types of items

- sidenotes⁽¹⁸¹⁾ and text box items⁽¹⁸³⁾ are printed in floating boxes.
- "page number" fields⁽¹⁸⁵⁾
- "page count" fields⁽¹⁸⁶⁾

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⁽²¹⁰⁾: SaveDocX⁽³³³⁾, SaveDocXToStream⁽³³⁴⁾.

New OnSaveDocXExtra⁽³⁹⁹⁾ event.

New options for TCustomRVFontInfo⁽⁶⁹⁶⁾.Options⁽⁷⁰⁴⁾: *rvteoDocXCode* and *rvteoDocXInRunCode*.

New properties of TRVOfficeConverter⁽⁷⁹⁵⁾: ExcludeDocXImportConverter, ExcludeDocXExportConverter⁽⁷⁹⁷⁾.

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⁽⁶²⁵⁾ supports a two finger zooming as well.

Selection handles are implemented in TRichView⁽²¹⁰⁾ and TRichViewEdit⁽⁴⁶¹⁾. 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⁽²⁵³⁾ property.

New property: TRVStyle.SelectionHandleKind⁽⁶⁴⁸⁾.

Printing and drawing

New properties defining sizes in TRVPrint⁽⁷⁵⁹⁾: Margins⁽⁷⁶⁸⁾, HeaderY⁽⁷⁶⁷⁾, FooterY⁽⁷⁶⁶⁾, Units⁽⁷⁷⁰⁾. 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⁽⁷⁷⁹⁾ and SetFooter⁽⁷⁷⁸⁾ have additional optional parameter, specifying header/footer type. New properties: TitlePage⁽⁷⁶⁹⁾, FacingPages⁽⁷⁶⁵⁾.

New property: PageNoFromNumber⁽⁸²⁶⁾ specifies the starting value for "page number" fields⁽¹⁸⁵⁾. In TRVReportHelper⁽⁸⁰⁴⁾, page numbers can be requested in OnGetPageNumber⁽⁸¹¹⁾ event.

New TRVReportHelper⁽⁸⁰⁴⁾.NoMetafiles property allows drawing metafiles as bitmaps.

Editing

New methods allowing to change properties of TRichViewEdit as an editing (undoable) operation: SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd⁽⁵⁴⁵⁾, SetBackgroundImageEd⁽⁵⁴⁶⁾. They allow changing margins, DocParameters, background properties.

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

- SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾, SetItemExtraIntPropertyExEd⁽⁵⁵⁶⁾;
- SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾, SetItemExtraStrPropertyExEd⁽⁵⁵⁷⁾.

They support more properties than the old methods.

Text attributes

New property SizeDouble⁽⁷⁰⁷⁾ allows defining font size with higher precision.

Visual changes

- New TRVStyle⁽⁶³⁰⁾.FontQuality⁽⁶³⁹⁾ property.
- Smooth image scaling⁽¹²⁰⁾.

Other changes

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

Footnotes⁽¹⁸⁰⁾/endnotes⁽¹⁷⁸⁾ displaying a character (instead of a number) are supported.

All work with graphic classes is encapsulated in TRVGraphicHandler⁽⁹⁷²⁾ class, available via RVGraphicHandler⁽¹⁰⁵⁷⁾ variable.

RichViewJustifyBeforeLineBreak⁽¹⁰⁴⁶⁾ variable allow to turn on justifying lines before line breaks.

Faster processing of hidden text.

New properties:

- TRVScroller⁽⁸¹³⁾.NoHScroll⁽⁸¹⁷⁾;
- TCustomRichViewEdit⁽⁴⁶¹⁾.ForceFieldHighlight⁽⁴⁷⁸⁾.

New option for TCustomRichView⁽²¹⁰⁾.Options⁽²⁴⁰⁾: *rvoPercentEncodedURL*.

2.10 New in version 14

Compatibility issues

- Item tags⁽⁹¹⁾ 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⁽³⁸⁸⁾, OnRVControlNeeded⁽³⁹²⁾). 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⁽³³³⁾ 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⁽³³⁰⁾, LoadTextFromStream⁽³³¹⁾, SaveText⁽³⁴⁵⁾, SaveTextToStream⁽³⁴⁵⁾, TRichViewEdit.InsertTextFromFile⁽⁵³³⁾ 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⁽³⁴⁶⁾ and TRichViewEdit.SearchText⁽⁵⁴³⁾). GetRVSearchOptions⁽⁹⁸⁷⁾ and GetRVESearchOptions⁽⁹⁸⁶⁾ 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 *rvoLoadBack* is in RVFOptions⁽²⁴⁷⁾, TDBRichView⁽⁵⁹⁴⁾ clears BackgroundBitmap⁽²²²⁾ before loading data from its field. If *rvoLoadBack* is in RVFOptions⁽²⁴⁷⁾ and FieldFormat⁽⁶¹³⁾=*rddbRVF*, TDBRichViewEdit⁽⁶⁰⁶⁾ clears BackgroundBitmap⁽²²²⁾ before loading data from its field.
- PNG images are saved to RTF as PNG images only if *rvtfSavePngAsPng* is included in RTFOptions⁽²⁴⁵⁾. It is included by default. However, for existing projects, you may need to set it manually.
- The declaration of TRVLongOperation⁽³⁸⁵⁾ type is changed.
- The following global variables were removed from RVTable.pas: RichViewTableGridStyle, RichViewTableGridStyle2, RichViewTableGridColor. They were replaced with properties of TRVStyle⁽⁶⁴⁰⁾.
- A new option *rvoShowGridLines* in TRichView.Options⁽²⁴⁰⁾ 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⁽⁶⁵²⁾ represent "real" text and paragraph styles. When style templates are used, an appearance of text and paragraphs are still defined in TextStyles⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾. 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⁽²⁵⁶⁾ = True.

TRichView.StyleTemplateInsertMode⁽²⁵⁴⁾ specifies how style templates affect insertion of RVF and RTF data.

When the collection of style templates has been modified, OnStyleTemplatesChange⁽⁴¹⁰⁾ occurs.

In TRichViewEdit, you can apply the specified style template to the selection using the methods: ApplyStyleTemplate⁽⁴⁹³⁾, ApplyTextStyleTemplate⁽⁴⁹⁴⁾, ApplyParaStyleTemplate⁽⁴⁹¹⁾. Style templates can be edited using ChangeStyleTemplates⁽⁵⁰⁰⁾ method.

The function RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ has an additional optional parameter: a name of a style template for using for a note character.

TRVStyle.MainRVStyle⁽⁶⁴⁷⁾ property allows using a single collection of style templates for several documents.

See the overview topic about style templates⁽⁹⁸⁾.

Rotation

Table cells can be rotated⁽⁹⁰⁴⁾ by 90°, 180°, or 270°.

A new method returning item coordinates: GetItemCoordsEx⁽²⁹⁹⁾; it takes cell rotation into account.

String tags

Since version 13.2, item tags⁽⁹¹⁾ 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⁽⁹⁰⁶⁾.

64-bit

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

Note: TRVOfficeConverter⁽⁷⁹⁵⁾ 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⁽⁶²⁵⁾ adjusts its colors (background, page border, shadow) according to the current style.

Text files

A new optional parameter for `TRichView.LoadText`⁽³³⁰⁾, `LoadTextFromStream`⁽³³¹⁾, `SaveText`⁽³⁴⁵⁾, `SaveTextToStream`⁽³⁴⁵⁾, `TRichViewEdit.InsertTextFromFile`⁽⁵³³⁾: `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*⁽¹⁷⁰⁾ and *hotspots*⁽¹⁷²⁾ is saved to RVF.

The parameters of `TRichView.OnRVFPictureNeeded`⁽³⁹³⁾ were changed. Now, this event is called not only for pictures⁽¹⁶³⁾ and hot-pictures⁽¹⁶⁶⁾, but also for tables⁽¹⁷³⁾ 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`⁽²⁴⁵⁾: `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`⁽³⁸⁸⁾/`OnWriteHyperlink`⁽⁴¹¹⁾ is not assigned.

HTML saving

`TRichView.SaveHTMLEx`⁽³³⁷⁾ can save stretched background images (previously, they were saved as centered).

`TRichView.SaveHTMLEx`⁽³³⁷⁾ saves list markers better (both in normal and `rvsoMarkersAsText`⁽¹⁰²⁰⁾ modes).

New "hidden" parameters in `TRichView.OnSaveImage2`⁽⁴⁰²⁾ event.

Default processing if `OnWriteHyperlink`⁽⁴¹¹⁾ is not assigned.

`SaveHTMLEx`⁽³³⁷⁾ 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`⁽⁶⁵³⁾ property.
- All adjacent hypertext items⁽⁹²⁾ having the same non-empty tags⁽⁹¹⁾ are highlighted below the mouse pointer; so visually they represent a single hyperlink.
- New properties⁽¹⁰⁰⁰⁾ for non-text items: background color, border width and color, spacing around the border.

- Footnote⁽¹⁸⁰⁾ and endnote⁽¹⁷⁸⁾ 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⁽⁵³⁹⁾.

New property: AcceptPasteFormats⁽⁴⁷⁰⁾ allows to restrict a list of formats that the editor can paste.

New property: DefaultPictureVAlign⁽⁴⁷⁵⁾ defines an alignment for pasted and dropped images.

DB components

If *rvfoLoadBack* is in RVFOptions⁽²⁴⁷⁾, TDBRichView⁽⁵⁹⁴⁾ clears BackgroundBitmap⁽²²²⁾ before loading data from its field.

If *rvfoLoadBack* is in RVFOptions⁽²⁴⁷⁾ and FieldFormat⁽⁶¹³⁾=*rvdbRVF*, TDBRichViewEdit⁽⁶⁰⁶⁾ clears BackgroundBitmap⁽²²²⁾ before loading data from its field.

Printing

Tables

New property for a table row: KeepTogether⁽⁹¹⁰⁾. New table methods: SetRowPageBreakBefore, SetRowKeepTogether⁽⁸⁸⁹⁾.

New option for table.PrintOptions⁽⁸⁶³⁾: *rvtoContinue*.

Other

New property: TCustomRVPrint.IgnorePageBreaks⁽⁸²³⁾.

Tables

New properties of TRVStyle defining the appearance of grid lines: GridColor, GridStyle, GridReadOnlyColor, GridReadOnlyStyle⁽⁶⁴⁰⁾. They replace global variables used in older versions. New option *rvShowGridLines* in TRichView.Options⁽²⁴⁰⁾.

Other changes

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

New event of TRichViewEdit⁽⁴⁶¹⁾: OnCheckStickingItems⁽⁵⁷³⁾ event allows to stick items together.

New typed constant: RichViewDoNotMergeNumbering⁽¹⁰⁴⁵⁾.

"Smart start" search option for TRichView.SearchText⁽³⁴⁶⁾ and TRichViewEdit.SearchText⁽⁵⁴³⁾.

TRichView.OnProgress⁽³⁸⁵⁾ 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⁽⁸⁰²⁾ and ExportRV⁽⁸⁰⁰⁾.

Better table layout algorithm.

New methods for TRVStyle: FindTextStyle, FindParaStyle⁽⁶⁶¹⁾, TextStyles.FindSuchStyleEx⁽⁶⁸¹⁾, ParaStyles.FindSuchStyleEx⁽⁶⁸⁵⁾.

2.11 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` property (*rvWrapSimple* value). But even with this value, a line breaking can be different for Unicode⁽¹³⁰⁾ text in paragraphs with `Alignment`⁽⁷¹⁹⁾ = *rvJustify*. `RichViewWrapAnywhere` (global constant) was removed, superseded by `TRVStyle.LineWrapMode`⁽⁶⁴⁴⁾ (*rvWrapAnywhere* value).
2. New parameter in `ApplyStyleConversion`⁽⁴⁹²⁾ and `ApplyParaStyleConversion`⁽⁴⁹¹⁾. 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 as pixels, so if they load such RVF files, all lengths will be too large.
4. This version defines widths of breaks as `TRVStyle.Length`⁽¹⁰²⁷⁾ instead of bytes. All applications that use `GetBreakInfo`⁽²⁸⁹⁾ and `GetCurrentBreakInfo`⁽⁵⁰⁴⁾ methods must correct the type of variable used to read a line width.
5. Paragraph borders are drawn slightly differently.

New line wrapping algorithm

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

New property `TRVStyle.LineWrapMode`⁽⁶⁴⁴⁾ allows changing a line wrapping mode.

Hidden items

Some items can be hidden⁽¹⁰¹⁾.

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.Units`⁽⁶³⁰⁾⁽⁶⁵⁵⁾ property. Instead of pixels, you can use twips⁽¹³⁹⁾.

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`⁽¹⁰⁴⁷⁾ = 0.

Shared item content

A new item property is added: `rvepShared`⁽¹⁰⁰⁰⁾. It allows for several picture items⁽¹⁶³⁾ to share the same graphic object. For a control item⁽¹⁶⁸⁾, it allows to reuse its control after the item deletion.

Incremental printing

TRVPrint⁽⁷⁵⁹⁾ and TRVReportHelper⁽⁸⁰⁴⁾ components have new properties: MinPrintedItemNo⁽⁸²⁵⁾ and MaxPrintedItemNo⁽⁸²⁴⁾. 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⁽⁴⁷⁴⁾ 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⁽⁴⁹²⁾ and ApplyParaStyleConversion⁽⁴⁹¹⁾ have the new parameter (Recursive) allowing to prevent processing of table cells.

Drawing and formatting

Improved table layout algorithm.

New option in RVVisibleSpecialCharacters⁽¹⁰⁶¹⁾: *rvscTab*. RVParagraphMarks⁽¹⁰⁵⁸⁾ global variable allows defining how paragraph breaks and line breaks are displayed in "show special characters" mode.

New property TRVStyle.DisabledFontColor⁽⁶³⁷⁾ allows change text color in disabled controls.

Calls of TRVStyle.OnDrawTextBack⁽⁶⁷³⁾ 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⁽⁸²⁸⁾ returns the page index for the given position in the document.

Saving and loading

Improved saving of footnotes⁽¹⁸⁰⁾ and endnotes⁽¹⁷⁸⁾ in plain text files.

Smarter replacing space characters to when saving HTML.

New text properties for TRichView.DocParameters⁽²³⁰⁾:

- Author⁽⁴¹⁶⁾,
- Title⁽⁴¹⁹⁾,
- Comments⁽⁴¹⁷⁾

Hypertext

rvNoReadOnlyJumps option from EditorOptions⁽⁴⁷⁵⁾ disallows automatic switching to a hypertext mode when TRichViewEdit becomes read-only⁽⁴⁸⁰⁾.

2.12 New in version 12

Compatibility issues

Assignment to `PageBreaksBeforeItems` ⁽²⁴⁴⁾ requires reformatting even if the given item starts a paragraph (because it clears text flow around side-aligned items).

`OutlineLevel` ⁽⁷²⁵⁾ 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 ⁽¹⁶¹⁾. While the new behavior (searching in multiple text items) requires specifying a new option in the parameter of `SearchText` methods explicitly, `GetRVSearchOptions` ⁽⁹⁸⁷⁾ and `GetRVESearchOptions` ⁽⁹⁸⁶⁾ functions always return this new option.

Left- and right-aligned objects

Two new values are available for `TRVAlign` ⁽¹⁰³³⁾ type: *rvvLeft* and *rvvRight*. They can be used for all non-text items, except for tables ⁽¹⁷³⁾, *breaks* ⁽¹⁶⁷⁾, tabs ⁽¹⁶²⁾, list markers ⁽¹⁷⁴⁾, footnotes ⁽¹⁸⁰⁾ and endnotes ⁽¹⁷⁸⁾.

New methods are available for accessing vertical alignment:

- `TRichView.GetItemVAlign` ⁽³⁰³⁾,
- `TRichView.SetItemVAlign` ⁽³⁶¹⁾,
- `TRichViewEdit.GetCurrentItemVAlign` ⁽⁵¹²⁾,
- `TRichViewEdit.SetItemVAlignEd` ⁽⁵⁶⁵⁾,
- `TRichViewEdit.SetCurrentItemVAlign` ⁽⁵⁶⁵⁾.

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

- `TRichView.ClearLeft` ⁽²²⁵⁾,
- `TRichView.ClearRight` ⁽²²⁶⁾,
- `TRichViewEdit.ClearTextFlow` ⁽⁵⁰¹⁾.

Showing invisible characters

New options for `RVVisibleSpecialCharacters` ⁽¹⁰⁶¹⁾ allowing to display/hide lines for floating objects, paragraph directions, printing options and outline levels ⁽⁷²⁵⁾.

New colors can be defined in `TRVStyle`:

- `FloatingLineColor` ⁽⁶³⁸⁾,
- `SpecialCharactersColor` ⁽⁶⁵²⁾.

Headings

New property in paragraph style: `OutlineLevel` ⁽⁷²⁵⁾ defines headings.

Other

New option of text searching (*rvsroMultiltem* for `TRichView.SearchText` ⁽³⁴⁶⁾, *rvseoMultiltem* for `TRichViewEdit.SearchText` ⁽⁵⁴³⁾) allows the searched string matching substrings of several text items.

New event in `TRVStyle`: `OnAfterApplyStyle` ⁽⁶⁶⁴⁾.

New event in TRichView: OnGetItemCursor⁽³⁷⁴⁾

New property in TRVLabelItemInfo: RemoveInternalLeading⁽⁹¹⁵⁾

New methods in TRVOfficeConverter: IsValidImporter⁽⁸⁰³⁾ allows checking a file before importing.

New values in TRichView.RVFOptions⁽²⁴⁷⁾ and RVFWarnings⁽²⁵¹⁾ (*rvfIgnoreUnknownCtrlProperties*, *rvfwUnknownCtrlProperties*)

2.13 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⁽⁹⁹³⁾, TRVUnicodeString⁽¹⁰³²⁾, TRVRawByteString⁽¹⁰¹⁹⁾. 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⁽¹⁰¹⁹⁾: OnDrawStyleText⁽⁶⁷¹⁾, OnReadHyperlink⁽³⁸⁸⁾, OnItemAction⁽³⁸⁰⁾, OnSaveItemToFile⁽⁴⁰⁴⁾, OnRVDbClick⁽³⁹¹⁾, OnRVRightClick⁽³⁹⁷⁾, OnItemTextEdit⁽⁵⁷⁸⁾. In existing projects, change types of parameters in these events manually.

In OnSaveComponentToFile⁽³⁹⁷⁾ 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⁽¹⁰¹⁹⁾ 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`⁽²⁷⁰⁾ (and its versions);
- `GetWordAt`⁽³¹⁴⁾;
- `GetItemText`⁽³⁰³⁾;
- `SetItemText`⁽³⁶⁰⁾;
- `GetCurrentItemText`⁽⁵¹¹⁾;
- `SetItemTextEd`⁽⁵⁶⁴⁾;
- `SetCurrentItemText`⁽⁵⁵⁷⁾.

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

- `CopyText`⁽²⁸³⁾;
- `GetSelText`⁽³¹³⁾;
- `InsertStringTag`⁽⁵³⁰⁾;
- `InsertText`⁽⁵³²⁾;
- `PasteText`⁽⁵³⁸⁾;
- `SearchText`⁽³⁴⁶⁾.

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

- `AddFmt`⁽²⁶⁶⁾;
- `AddItem`⁽²⁶⁹⁾;
- `InsertItem`⁽⁵²²⁾;
- `GetTextInfo`⁽³¹³⁾;
- `GetCurrentTextInfo`⁽⁵¹⁴⁾.

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`)⁽¹³⁰⁾;
- removing text formatting⁽¹⁰⁷¹⁾.

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⁽⁹¹⁾ (in `tags-are-PChars` mode);
- extra item properties⁽¹⁰⁰⁵⁾;
- names of *checkpoints*⁽⁸⁷⁾;
- visible text in labels⁽¹⁷⁶⁾, numbered sequences⁽¹⁷⁷⁾, endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾;
- text in list markers⁽¹⁷⁴⁾ (not only bullets, but all numbering);
- live spelling⁽¹³²⁾ interface;
- hypertext targets (`OnReadHyperlink`⁽³⁸⁸⁾, `OnWriteHyperlink`⁽³⁸⁸⁾);
- and others.

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

- Unicode⁽⁷¹⁵⁾ property of text style (from False to True);
- TRichView.RTFReadProperties⁽²⁴⁶⁾.UnicodeMode⁽⁴⁵⁹⁾ (from *rvruNoUnicode* to *rvruOnlyUnicode*);
- TRichView.Options⁽²⁴⁰⁾ (*rvAutoCopyUnicodeText* is included, *rvAutoCopyText* 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⁽⁷¹⁵⁾ property of text style is saved. When saving text styles (in RVF files⁽¹²⁴⁾ or Delphi forms) in Delphi/C++Builder 2009, value of Unicode⁽⁷¹⁵⁾ 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⁽⁶⁷¹⁾, OnReadHyperlink⁽³⁸⁸⁾, OnItemAction⁽³⁸⁰⁾, OnSaveItemToFile⁽⁴⁰⁴⁾, OnRVDbClick⁽³⁹¹⁾, OnRVRightClick⁽³⁹⁷⁾, OnItemTextEdit⁽⁵⁷⁸⁾) must be converted to String before using. For example, for OnRVDbClick⁽³⁹¹⁾ event:

```
// Example for Delphi 2009 or newer!!!
procedure TForm1.RichView1RVDbClick(Sender: TCustomRichView(210);
  ClickedWord: TRVRawByteString(1019); Style: Integer);
var s: String;
begin
  if (Style>=0) and Sender.Style(253).TextStyles(654)[Style].Unicode(715) then
    // converting from "raw Unicode" to UnicodeString
    s := RVU_RawUnicodeToWideString(992)(ClickedWord)
  else
    // converting from ANSI to UnicodeString
    s := RVU_RawUnicodeToWideString(992)( /
      RVU_AnsiToUnicode(992)(Sender.RVData(247).GetStyleCodePage(Style),
      ClickedWord));
    Panell1.Caption := s;
end;
```

New methods, properties and events

New TCustomRichView.OnAddStyle⁽³⁷⁰⁾ event.

New NoVScroll⁽⁸¹⁷⁾ property of TRichView.

New options for line spacing⁽⁷²³⁾ ("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⁽²⁴⁵⁾.

New properties for RTFReadProperties⁽²⁴⁶⁾: CharsetForUnicode⁽⁴⁵²⁾ and UseCharsetForUnicode⁽⁴⁵⁹⁾.

RVF⁽¹²⁴⁾:

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.

2.14 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⁽³³⁷⁾ 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⁽⁴⁰⁴⁾ 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⁽¹⁰⁴⁹⁾ to *False* to disable.

Editing

- BeginUpdate⁽⁴⁹⁸⁾ and EndUpdate⁽⁵⁰³⁾ have new functionality.
- Simple version GetWordAt⁽³¹⁴⁾ always returns ANSI string.
- Paste⁽⁵³⁴⁾ can paste graphic files in CF_HDROP format.

RTF

- bookmarks are read from RTF as checkpoints⁽⁸⁷⁾.
- Local links are converted automatically when reading and writing RTF.
- Sub/superscripts and VShift⁽⁷¹⁰⁾ are saved and loaded to RTF differently.

Other

- meaning of ItemNo parameter in OnControlAction⁽³⁷³⁾ is changed.
- RVF insertion methods ignore background.

Main new features

- Animation of pictures⁽¹¹⁹⁾;
- "Smart popups"⁽⁵⁸⁸⁾;
- page breaks inside table cells.

New item types⁽⁸⁵⁾

- labels⁽¹⁷⁶⁾;
- sequences⁽¹⁷⁷⁾;
- footnotes⁽¹⁸⁰⁾;
- endnotes⁽¹⁷⁸⁾;

- references to parent notes ⁽¹⁸⁴⁾.

New in saving and loading

HTML:

New options in Options parameter of SaveHTML ⁽³³⁶⁾ and SaveHTMLEx ⁽³³⁷⁾:

- `rvsoXHTML` ⁽¹⁰²⁰⁾ – if included, XHTML is saved instead of HTML;
- `rvsoUTF8` ⁽¹⁰²⁰⁾ – if included, HTML/XHTML is saved in UTF-8 encoding.

You can choose between using `<p>` and `<div>` ⁽¹⁰⁴⁰⁾.

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 ⁽²³⁰⁾ property.

Options ⁽²⁴⁵⁾ for saving smaller RTF files: `rvrtfSavePicturesBinary`, `rvrtfPNGInsteadOfBitmap`.

RTF bookmarks are read as *checkpoints* ⁽⁸⁷⁾. 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 ⁽²⁴⁶⁾:

- StoreImagesFileNames allows storing file names of external images (update: this property is removed in version 16);
- BasePathLinks ⁽⁴⁵¹⁾ specifies whether to add path to file in hyperlinks;
- IgnoreBookmarks ⁽⁴⁵⁴⁾ allows ignoring bookmarks;
- IgnoreNotes ⁽⁴⁵⁵⁾ allows ignoring footnotes and endnotes;
- IgnoreSequences ⁽⁴⁵⁵⁾ allows ignoring numbered sequences;
- IgnoreTables ⁽⁴⁵⁵⁾ allows ignoring tables;
- ReadDocParameters ⁽⁴⁵⁷⁾ allows reading DocParameters;
- UseSingleCellPadding ⁽⁴⁶⁰⁾ sets mode for loading tables cell padding.

Text:

- You can allow saving explicit page breaks in text files/streams. Set RichViewSavePageBreaksInText ⁽¹⁰⁴⁰⁾ typed constant to *True*.

Other:

- LoadFromStream ⁽³²¹⁾ loads document in RVF, RTF or text formats (autodetect).

New in DB controls

- TDBRichView ⁽⁵⁹⁴⁾ and TDBRichViewEdit ⁽⁶⁰⁶⁾ can be placed in TDBCtrlGrid.
- TDBRichView ⁽⁵⁹⁴⁾ and TDBRichViewEdit ⁽⁶⁰⁶⁾ can update standard DB actions.
- TDBRichView ⁽⁵⁹⁴⁾.OnLoadCustomFormat ⁽³⁸³⁾, TDBRichViewEdit ⁽⁶⁰⁶⁾.OnLoadCustomFormat and OnSaveCustomFormat events allow loading and saving data to fields in custom formats.
- Special support for Unicode memo fields.

New in drag&drop ⁽¹¹⁸⁾

Low level events:

- `OnOleDragEnter`⁽⁵⁷⁹⁾;
- `OnOleDragOver`⁽⁵⁸⁰⁾;
- `OnOleDragLeave`⁽⁵⁷⁹⁾;
- `OnOleDrop`⁽⁵⁸¹⁾.

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`⁽³⁴⁶⁾ and `TCustomRichViewEdit.SearchTextW`⁽⁵⁴³⁾ allows searching for `WideString`.

New in layout and formatting

- If you set `RichViewShowGhostSpaces`⁽¹⁰⁴⁸⁾ typed constant to `True`, spaces hidden between lines will be shown if `rvShowSpecialCharacters` is in `TCustomRichView.Options`⁽²⁴⁰⁾.
- You can define which characters to show in `ShowSpecialCharacters`⁽²⁴⁰⁾ mode. They are listed in `RVVisibleSpecialCharacters`⁽¹⁰⁶¹⁾ typed constant.
- New possible values for `TCustomRichView.BackgroundStyle`⁽²²³⁾.
- New property `TCustomRichView.MaxLength`⁽²³⁹⁾ – maximal count of characters per line.
- New property `TCustomRichView.WordWrap`⁽²⁵⁸⁾ 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⁽⁵⁷⁾).

New text attributes

- New text Protection⁽⁷⁰⁵⁾ options: `rvprStyleSplitProtect`, `rvprSticking2`, `rvprSticking3`.
- New property `HoverEffects`⁽⁷⁰³⁾ allows hyperlinks underlining.
- New properties `UnderlineType`⁽⁷⁰⁹⁾, `UnderlineColor`⁽⁷⁰⁹⁾ and `HoverUnderlineColor`⁽⁷⁰³⁾ allow to customize underlines.
- New property `SubSuperScriptType`⁽⁷⁰⁸⁾ allows implementing subscripts and superscripts.

New in printing

- `TRVReportHelper`⁽⁸⁰⁴⁾ and `TRVPrint`⁽⁷⁵⁹⁾ do not require parent control any more. They can be placed on data module or created without owner.
- `TRVPrintPreview`⁽⁶²⁵⁾ displays full paper size, including areas where printer cannot print (see WinAPI function `GetDeviceCaps`, `PHYSICAL***` constants).
- `TRVPrintPreview.PrintableAreaPen`⁽⁸³⁸⁾ 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`⁽⁷⁶⁶⁾ property.

New in item properties

- `rvepVisible` item property⁽¹⁰⁰⁰⁾ allows to hide controls⁽¹⁶⁸⁾;
- `rvepResizable` item property⁽¹⁰⁰⁰⁾ works for pictures⁽¹⁶³⁾ too;
- `rvespAlt` item property⁽¹⁰⁰⁵⁾ can be applied to *bullets*⁽¹⁷⁰⁾ and *hotspots*⁽¹⁷²⁾ as well as to pictures⁽¹⁶³⁾ and *hot-pictures*⁽¹⁶³⁾;

- new values for `VAlign` ⁽¹⁰³³⁾ property of items: `rvvaAbsTop`, `rvvaAbsBottom`, `rvvaAbsMiddle`.
- new styles ⁽⁹⁹⁵⁾ for `breaks` ⁽¹⁶⁷⁾ (horizontal lines): dashed and dotted; all types of `breaks` are drawn without rounded corners.

New in checkpoints ⁽⁸⁷⁾

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

- `InsertCheckpoint` ⁽⁵¹⁶⁾;
- `GetCheckpointAtCaret` ⁽⁵⁰³⁾;
- `RemoveCheckpointAtCaret` ⁽⁵⁴¹⁾.

New in tables ⁽¹⁷³⁾

- Table cells can be printed on several pages (previously, page breaks were possible only between rows).
- New option for `table.Options` ⁽⁸⁶¹⁾: `rvtoNoCellSelect`. If included, multicell selection by mouse is not allowed.
- New property `table.VisibleBorders` ⁽⁸⁶⁵⁾ allows to hide some sides of table border. Direct assignment to this property cannot be undone/redone, use `table.SetTableVisibleBorders` ⁽⁸⁸⁹⁾ 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 ⁽¹²⁴⁾ 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` ⁽¹⁰⁴⁹⁾ to `False` to disable this feature).
- New properties `RowCount` ⁽⁸⁶⁴⁾, `ColCount` ⁽⁸⁵⁸⁾.
- New properties `CellHPadding` ⁽⁸⁵⁵⁾, `CellVPadding` ⁽⁸⁵⁷⁾.
- New property `CellHint` ⁽⁸⁹⁵⁾.
- New property `RowPageBreakBefore` ⁽⁹⁰⁹⁾.
- New methods `SetCellBackgroundImageFileName` ⁽⁸⁸⁵⁾ and `SetCellHint` ⁽⁸⁸⁷⁾.
- Improved table layout algorithm.
- better `SplitSelectedCellsVertically` ⁽⁸⁹⁰⁾ and `SplitSelectedCellsHorizontally` ⁽⁸⁹⁰⁾ implementations.

New in RVF ⁽¹²⁴⁾

If image was not returned in `OnRVFPictureNeeded` ⁽³⁹³⁾ event, this event is called the second time with the value of `rvsplImageFileName` ⁽¹⁰⁰⁵⁾ 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` ⁽⁴⁷⁴⁾, measured in `OnMeasureCustomCaret` ⁽⁵⁷⁸⁾ event, drawn in `OnDrawCustomCaret` ⁽⁵⁷⁵⁾ 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⁽⁵³⁵⁾.
- New characters are included in the default value of TCustomRichView.Delimiters⁽²²⁸⁾.
- meaning of TCustomRichViewEdit's BeginUpdate⁽⁴⁹⁸⁾ and EndUpdate⁽⁵⁰³⁾ was changed.
- New option for TCustomRichView.SearchText⁽³⁴⁶⁾: *rvsroFromStart*.
- Changes in TCustomRichView.GetWordAt⁽³¹⁴⁾.
- Ability to delete unused styles in multiple documents, see TRVDeleteUnusedStylesData⁽⁹⁶⁹⁾.
- Much faster reformatting, especially if *rvoFastFormatting* is included in TCustomRichView.Options⁽²⁴⁰⁾ (default). Much faster loading RTF files. Much faster copying to the Clipboard.
- Function RVIsCustomURL⁽¹⁰⁵⁸⁾.
- **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⁽³⁷³⁾.

2.15 New in version 1.9

Compatibility issues

- TRVStyle.CurrentItemColor⁽⁶³⁵⁾ is used for drawing frame around the current item instead of TRVStyle HoverColor⁽⁶⁴¹⁾.
- ApplyTextStyle⁽⁴⁹⁴⁾, ApplyParaStyle⁽⁴⁹⁰⁾, ApplyStyleConversion⁽⁴⁹²⁾, ApplyParaStyleConversion⁽⁴⁹¹⁾ are applied to cells of all selected tables.
- OnSaveImage2⁽⁴⁰²⁾ event: ImageSaveNo is declared as a var-parameter (it does not make sense otherwise)
- RTF export: only Standard⁽⁶⁹⁵⁾ styles can be exported as style sheet.
- Default value of RVStyle.SpacesInTab⁽⁶⁵²⁾ is changed from 8 to 0.
- Default value of TRichViewEdit.EditorOptions⁽⁴⁷⁵⁾ is changed from [rvoClearTagOnStyleApp] to [rvoWantTabs, rvoCtrlJumps].
- New characters are added to default value of Delimiters⁽²²⁸⁾ property.

New: tab stops

- Tabs⁽⁷²⁶⁾ – properties of paragraph style defining tab stops.
- TRVStyle.DefTabWidth⁽⁶³⁶⁾ defines default distance between tab stops.
- TRVStyle.SpacesInTab⁽⁶⁵²⁾: if equals to 0, methods for adding/inserting multiline text, RTF, and Tab key inserts tabulator⁽¹⁶²⁾ (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⁽²⁷³⁾, TRichViewEdit.InsertTab⁽⁵³¹⁾ adds/inserts tabulator⁽¹⁶²⁾

New: Live spelling check

See overview topic ⁽¹³²⁾

New in TRichView

- VAlign ⁽²⁵⁷⁾ property, ClientToDocument ⁽²⁸⁰⁾ method;
- *rvoShowSpecialCharacters* in Options ⁽²⁴⁰⁾;
- BeginOleDrag ⁽²⁷⁸⁾ – allows drag&drop of inserted controls;
- Reformat ⁽³³²⁾ method – formatting method that keeps selection;
- AddTextNLA ⁽²⁷⁴⁾ method;
- LoadTextFromStream, LoadTextFromStreamW ⁽³³¹⁾;
- Events: OnItemHint ⁽³⁸¹⁾, OnProgress ⁽³⁸⁵⁾.

New item properties

- additional string properties ⁽¹⁰⁰⁵⁾: hint, alt text, image file name; a set of methods supporting them: TRichView's GetItemExtraStrProperty ⁽³⁰⁰⁾ and SetItemExtraStrProperty ⁽³⁵⁹⁾, TRichViewEdit's GetCurrentItemExtraStrProperty ⁽⁵¹⁰⁾, SetCurrentItemExtraStrProperty ⁽⁵⁵⁷⁾ and SetItemExtraStrPropertyEd ⁽⁵⁶²⁾ (this set of methods is similar to methods working with additional integer properties ⁽¹⁰⁰⁰⁾);
- *rveoNoHTMLImagesSize* additional integer property ⁽¹⁰⁰⁰⁾

Printing

Added: *rvpaoKeepLinesTogether* and *rvpaoKeepWithNext* Options ⁽⁷²³⁾ for paragraph style.

Tables

- *rvtolgnoreContentWidth* and *rvtolgnoreContentHeight* options for TRVTableItemInfo ⁽⁸⁴⁶⁾.Options ⁽⁸⁶¹⁾;
- AssignProperties ⁽⁸⁷⁰⁾ method;
- better table layout algorithm.

RTF

- TRichView.RTFReadProperties ⁽⁴⁵⁰⁾.ExtractMetafileBitmaps ⁽⁴⁵⁴⁾ – 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 ⁽³³⁷⁾ is modified to conform "HTML 4.01 Transitional" standard, see <http://validator.w3.org/>;

- improvements in CSS export;
- SaveHTMLEx⁽³³⁷⁾ saves characters of "Symbol" font as HTML entities (like α);
- better saving of RTF-like tables (with CellHSpacing⁽⁸⁵⁶⁾=CellVSpacing⁽⁸⁵⁷⁾=-1 to simulate 1-pixel width border) with SaveHTMLEx⁽³³⁷⁾, or SaveHTML⁽³³⁶⁾ with *rvsoForceNonTextCSS* in the Options parameter;
- ImagesPrefix parameter of SaveHTML⁽³³⁶⁾/SaveHTMLEx⁽³³⁷⁾ can contain a full path (determined by the presence of ':').

TRVOfficeConverter⁽⁷⁹⁵⁾

- added: PreviewMode⁽⁷⁹⁹⁾ property.

TRVReportHelper⁽⁸⁰⁴⁾

- added: OnDrawCheckpoint⁽⁸³⁰⁾ event

Database controls

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

Other improvements

- TRVStyle.CurrentItemColor⁽⁶³⁵⁾.
- *rvNoCaretHighlightJumps* in EditorOptions⁽⁴⁷⁵⁾;
- faster redrawing on typing;
- better processing of bidirectional text⁽¹³²⁾;
- ApplyTextStyle⁽⁴⁹⁴⁾, ApplyParaStyle⁽⁴⁹⁰⁾, ApplyStyleConversion⁽⁴⁹²⁾, ApplyParaStyleConversion⁽⁴⁹¹⁾ are applied to cells of all selected tables;
- all TRichViewEdit methods for inserting one item⁽¹¹⁴⁾ 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 *rvShowSpecialCharacters* is in Options,⁽²⁴⁰⁾ 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⁽³³²⁾ 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.

2.16 New in version 1.8

Compatibility issues

Changes that can affect existing code:

- OnHTMLSaveImage⁽³⁷⁵⁾ is called for tables to save background image;
- OnHTMLSaveImage⁽³⁷⁵⁾ 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⁽²³⁹⁾ property does not include left and right margins any more, so now it works according to the help file⁽¹⁰⁵⁾;
- parameters are changed in OnChanging⁽⁵⁷²⁾ event (should not be a problem because this event is rarely used);
- table.OnDrawBorder⁽⁸⁹³⁾ event has two new parameters: Row and Col;
- changes in *rvteoHTMLCode* and *rvteoRTFCode* (Options⁽⁷⁰⁴⁾ 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⁽⁶⁴⁹⁾;
- RVF version number is changed from 1.2 to 1.3 (see RVF specification⁽¹⁴³⁾); 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*⁽⁷⁰⁵⁾ protection option meaning was changed (split into 3 *rvprStick**** options).

Drag&Drop

See overview⁽¹¹⁸⁾.

Background images for tables and cells

- table.BackgroundImage⁽⁸⁴⁹⁾, BackgroundStyle⁽⁸⁵⁰⁾;
- cell.BackgroundImage⁽⁸⁹⁶⁾, BackgroundStyle⁽⁸⁹⁷⁾.

Resizing pictures and controls with mouse

By default, all pictures⁽¹⁶³⁾ and hot-pictures⁽¹⁶⁶⁾ can be resized with the mouse (except for TIcon-s).

By default, controls⁽¹⁶⁸⁾ cannot be resized. But you can explicitly allow resizing for the given control, if you set its *rvepResizable*⁽¹⁰⁰⁰⁾ property to non-zero value. See also SelectControl⁽³⁴⁸⁾ method.

You can disallow resizing in the given RichViewEdit by including *rvoNoImageResize* in its EditorOptions⁽⁴⁷⁵⁾ property.

Extra items' integer properties⁽¹⁰⁰⁰⁾

- *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`⁽⁸²²⁾ property for `TRVPrint`⁽⁷⁵⁹⁾ and `TRVReportHelper`⁽⁸⁰⁴⁾: allows black and white or grayscale printing/previewing for color documents;
- much more accurate print preview⁽⁶²⁵⁾;
- new properties of `TRVPrintPreview`: `ClickMode`⁽⁸³⁶⁾, `Color`⁽⁸³⁶⁾, `PageBorderColor`⁽⁸³⁷⁾, `PageBorderWidth`⁽⁸³⁸⁾, `ShadowColor`⁽⁸³⁹⁾, `ShadowWidth`⁽⁸³⁹⁾.

Selecting:

- new line selection mode, see `TRVStyle.LineSelectCursor`⁽⁶⁴³⁾
- defining visual appearance of selection, see `TRVStyle.SelectionStyle`⁽⁶⁴⁹⁾
- defining selection mode, see `TRVStyle.SelectionMode`⁽⁶⁴⁹⁾

RVF⁽¹²⁴⁾

- 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`⁽²³¹⁾).

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`⁽⁴⁰¹⁾ event now allows to insert HTML code before the `</body>` tag;
- new `OnSaveParaToHTML`⁽⁴⁰⁶⁾, `OnSaveImage2`⁽⁴⁰²⁾, `OnSaveItemToFile`⁽⁴⁰⁴⁾ events;
- new options⁽¹⁰²⁰⁾ for HTML export;
- `OnWriteHyperlink`⁽⁴¹¹⁾ event supersedes `OnURLNeeded`.

RTF export

- `OnWriteHyperlink`⁽⁴¹¹⁾ event supersedes `OnURLNeeded`;
- new `OnSaveItemToFile`⁽⁴⁰⁴⁾ 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`⁽³⁷⁸⁾

DB controls

- new event: TDBRichViewEdit.OnNewDocument;
- new property: TDBRichViewEdit.IgnoreEscape⁽⁶¹⁴⁾.

Other improvements

- TRVReportHelper.OnDrawHyperlink⁽⁸³⁰⁾ event;
- default text styles for paragraph styles (TParaInfo.DefStyleNo⁽⁷²⁹⁾);
- improvements in processing of bidirectional text⁽¹³²⁾;
- new action in OnItemAction⁽³⁸⁰⁾ event: *rvalInserted*;
- faster redrawing on editing;
- previously reserved break styles⁽⁹⁹⁵⁾ are implemented;
- RefreshListMarkers⁽³³²⁾ method.

2.17 New in version 1.7

Compatibility issues

- TCustomRichView.DeleteLines is renamed to DeleteItems⁽²⁸³⁾.
- If you assign non-default values to extra item properties⁽¹⁰⁰⁰⁾, saved RVF files will not be opened with applications that use older version of RichView.
- OnDrawPageBreak⁽⁶⁶⁸⁾ 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⁽³⁹³⁾ was changed; if you already use this event, you must change declaration of event handler manually;
- FirstIndent⁽⁷²²⁾ is not applied to line starts inside paragraphs (**Shift + Enter**).
- DeleteUnusedStyles⁽²⁸⁶⁾ has a new parameter (deleting list styles).
- GetWordAt⁽³¹⁴⁾ parameters are changed.
- Accessing invalid item in RVStyle.TextStyles and ParaStyles does not cause "List index is out of bounds" error any more.
- OnHTMLSaveImage⁽³⁷⁵⁾ is called for background bitmap⁽²²²⁾; 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⁽⁶⁴⁶⁾

Methods of TCustomRichView:

- SetListMarkerInfo⁽³⁶²⁾,
- GetListMarkerInfo⁽³⁰⁷⁾.

Methods of TCustomRichViewEdit:

- ApplyListStyle⁽⁴⁹⁰⁾,
- RemoveLists⁽⁵⁴²⁾,
- ChangeListLevels⁽⁵⁰⁰⁾.

New options⁽¹⁰²⁰⁾ for HTML export – *rvsoMarkersAsText*.

New parameter in DeleteUnusedStyles⁽²⁸⁶⁾ (deleting list styles)

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

Methods of TCustomRichView:

- SetItemExtraIntProperty⁽³⁵⁸⁾ – changes value of property (without document reformatting);
- GetItemExtraIntProperty⁽³⁰⁰⁾ – returns value of property.

Methods of TCustomRichViewEdit:

- SetItemExtraIntPropertyEd⁽⁵⁶²⁾ – changes value of property, updates the document;
- SetCurrentItemExtraIntProperty⁽⁵⁵⁶⁾ – the same for item at the position of caret;
- GetCurrentItemExtraIntProperty⁽⁵¹⁰⁾ – returns value of property of item at the position of caret.

Soft page breaks (preliminary implementation)

Methods of TCustomRichView:

- AssignSoftPageBreaks⁽²⁷⁷⁾, ClearSoftPageBreaks⁽²⁷⁹⁾ - displays/hides soft page breaks

Property of TRVStyle

- SoftPageBreakColor⁽⁶⁵¹⁾

Other improvements in TCustomRichView

New methods:

- DeleteParas⁽²⁸⁵⁾ – deletes the specified paragraphs, updates the document;
- GetLineNo⁽³⁰⁶⁾ – returns line for the specified position;
- GetWordAt⁽³¹⁴⁾ (parameters are changed);
- GetItemAt⁽²⁹⁷⁾ – returns item in the given coordinates.

New events:

- OnItemAction⁽³⁸⁰⁾ – occurs on operations with item;
- OnPaint⁽³⁸⁴⁾.

New properties:

- RTFReadProperties.ConvertHighlight⁽⁴⁵²⁾, LineBreaksAsParagraphs⁽⁴⁵⁶⁾.

New Options⁽¹⁰²⁰⁾ for HTML export: *rvsoUseCheckpointsNames*.

New Options⁽²⁴⁷⁾ for RVF: *rvfoSaveLayout*, *rvfoLoadLayout*.

Events:

- OnSaveHTMLExtra⁽⁴⁰¹⁾, OnSaveRTFExtra⁽⁴⁰⁶⁾ – allow saving additional information in HTML and RTF.

Other improvements in TCustomRichViewEdit

Methods:

- ApplyParaStyleConversion⁽⁴⁹¹⁾ (and OnParaStyleConversion⁽⁵⁸²⁾ event) – implements commands like "change paragraph alignment", "increase indent", etc.
- InsertText, InsertTextW⁽⁵³²⁾ have additional optional parameter;
- GetCurrentLineCol⁽⁵¹²⁾ – returns line and column at the caret position;
- ConvertToPicture⁽⁵⁰²⁾, ConvertToHotPicture⁽⁵⁰¹⁾ – convert picture to *hot-picture* and vice versa

Properties:

- Modified⁽⁴⁷⁹⁾: Boolean

New EditorOption⁽⁴⁷⁵⁾ – *rvoDoNotWantShiftReturns*.

Other improvements in TRVStyle⁽⁶³⁰⁾

- InvalidPicture⁽⁶⁴³⁾ – picture that used instead of invalid or missed pictures;
- InvalidItem⁽⁶⁷⁶⁾ for TextStyles, InvalidItem⁽⁶⁸³⁾ for ParaStyles – are returned when accessing item which is out of index range.

New properties of text style (TFontInfo⁽⁷¹²⁾):

- Options⁽⁷⁰⁴⁾ – affects HTML and RTF export

New properties of paragraph style (TParaInfo⁽⁷¹⁸⁾):

- new protection Option⁽⁷²³⁾ – *rvpaoDoNotWantReturns*.

Improvements in TDBRichViewEdit⁽⁶⁰⁶⁾

- AutoDeleteUnusedStyles⁽⁶¹¹⁾

Improvements in TRVOfficeConverter⁽⁷⁹⁵⁾

- ExcludeHTMLImportConverter, ExcludeHTMLExportConverter⁽⁷⁹⁷⁾;
- ExtensionsInFilter⁽⁷⁹⁸⁾;
- ErrorCode⁽⁷⁹⁶⁾ – returns error code for the last import/export operation.

Improvements in TRVPrint⁽⁷⁵⁹⁾

- OnPrintComponent⁽⁸³¹⁾ is optional since this version;
- ClipMargins⁽⁷⁶⁵⁾ property – disallows printing on margins;
- headers and footers: TRVPrint.SetHeader⁽⁷⁷⁹⁾, SetFooter⁽⁷⁷⁸⁾, 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⁽³⁵⁶⁾, SetFooter⁽³⁵⁵⁾, DocParameters⁽²³⁰⁾).
- OnPagePostpaint⁽⁷⁸¹⁾;
- ColorMode⁽⁸²²⁾ – allows black&white and grayscale printing and preview;

Improvements in all visual controls of the package

- UseXPThemes⁽⁸¹⁸⁾ - allows using WindowsXP themes (visual styles)

Improvements in tables⁽¹⁹⁰⁾

Global variable RichViewTableGridStyle: TPenStyle – drawing invisible table borders.

Methods:

- SetCellVisibleBorders⁽⁸⁸⁹⁾ – sets VisibleBorders property for the cell.

Properties:

- HeadingRowCount⁽⁸⁵⁹⁾ – defines table header;
- new Option⁽⁸⁶¹⁾ – *rvtoRTFAllowAutofit*.

Events:

- OnCellEditing⁽⁸⁹²⁾ – allows to prevent cell editing;
- OnDrawBorder⁽⁸⁹³⁾ – custom drawing for cell borders.

New help topics

- Valid Documents⁽¹³⁸⁾
- Custom Drawing⁽¹⁴¹⁾
- Examples for DBRichViewEdit⁽⁶⁰⁶⁾
- New overview: Item Types⁽¹⁵⁸⁾
- 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.

2.18 New in version 1.6

Compatibility issues

- TRVPrint.OnPrintComponent⁽⁸³¹⁾ 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⁽²⁸⁶⁾.

Changes in TRVPrint

- new: TRVPrint.TransparentBackground⁽⁸²⁷⁾
- new: TRVPrint.GetFirstItemOnPage⁽⁸²⁷⁾
- new: TRVPrint.MirrorMargins⁽⁷⁶⁹⁾

Changes in tables

- Vertical alignment of cells (cells' `VAlign`⁹⁰⁶ and rows' `VAlign`⁹¹⁰, `SetCellVAlign`⁸⁸⁸ and `SetRowVAlign`⁸⁸⁹ of table).
- `BorderColor`⁸⁹⁹ and `BorderLightColor`⁸⁹⁹ of cells allow to override default values (see also `SetCellBorderColor`⁸⁸⁶ and `SetCellBorderLightColor`⁸⁸⁶).
- Ability to hide table grid in editor (new option for table `Options`⁸⁶¹ – `rvtoHideGridLines`).
- Ability to print table background in white (new option for table `PrintOptions`⁸⁶³ – `rvtoWhiteBackground`).
- New table `Options`⁸⁶¹ – `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`⁸⁶¹). Previous versions show grid only for zero-width borders.

New components



`TRVOfficeConverter`⁷⁹⁵ – allows using Microsoft Office converters



`TRVReportHelper`⁸⁰⁴ – reserved for future use.

HTML export

New options¹⁰²⁰ – `rvsolImageSizes`, `rvsoForceNonTextCSS`.

RTF export

You can save hyperlinks in RTF, using `OnURLNeeded` event (update: this event is removed in v17; use `OnWriteHyperlink`⁴¹¹).

New option²⁴⁵ for saving RTF files – `rvrtfSaveEMFDefault`.

RTF import

- new: reading hyperlinks: `OnReadHyperlink`³⁸⁸ event;
- reading tables (see also: `RTFReadProperties.AutoHideTableGridLines`⁴⁵¹);
- reading external images (see important note for using third-party graphics formats¹⁰⁶⁹);
- new: ignoring hidden text (see `RTFReadProperties.SkipHiddenText`⁴⁵⁸).

Horizontal scrolling

new: `HScrollPos`⁸¹⁶, `HScrollMax`⁸¹⁶, `OnHScrolled`⁸²¹.

Visual effects and formatting

- new: `TRVStyle.InactiveSelColor` and `InactiveSelTextColor`⁶⁴¹;

- new: CharSpacing⁽⁷⁰⁰⁾ property of text style.

DB Components

DB components can display RTF and plain text fields.

new: TDBRichViewEdit.FieldFormat⁽⁶¹³⁾ 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⁽⁵³⁰⁾.
- Parameters are added in TRichView.DeleteUnusedStyles⁽²⁸⁶⁾.
- Essential styles can be marked with Standard⁽⁶⁹⁵⁾ property.
- TRichView.SearchText⁽³⁴⁶⁾ and TRichViewEdit.SearchText⁽⁵⁴³⁾ now searches in table cells.
- Hypertext works automatically in read-only⁽⁴⁸⁰⁾ editors (user does not need to press and hold **Ctrl**).
- Editor can hide caret in read-only mode (new option in EditorOptions⁽⁴⁷⁵⁾ - *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⁽⁵⁴⁵⁾.

2.19 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 ⁽³²⁾.

3. New parameter

New parameter of SaveCSS ⁽⁶⁶²⁾ / SaveCSSToStream ⁽⁶⁶³⁾. Please use empty set ([]) for it.

New feature: RTF Import

- TRichView.LoadRTF ⁽³²⁶⁾;
- TRichView.LoadRTFFromStream ⁽³²⁷⁾;
- TRichViewEdit.InsertRTFFromFileEd ⁽⁵²⁷⁾;
- TRichViewEdit.InsertRTFFromStreamEd ⁽⁵²⁸⁾;
- TRichViewEdit.PasteRTF ⁽⁵³⁷⁾;
- TRichViewEdit.CanPasteRTF ⁽⁴⁹⁹⁾;
- and property TRichView.RTFReadProperties ⁽²⁴⁶⁾.

New feature: saving text and paragraph styles in RVF

- New *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions ⁽²⁴⁷⁾
- TRichView.RVFTextStylesReadMode ⁽²⁵¹⁾;
- TRichView.RVFPaStylesReadMode ⁽²⁵¹⁾;
- Related: TRichView.DeleteUnusedStyles ⁽²⁸⁶⁾.

New feature: hot pictures

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

New methods: AddHotPicture ⁽²⁶⁷⁾, AddHotPictureTag ⁽²⁶⁷⁾, InsertHotPicture ⁽⁵¹⁹⁾.

Old methods (GetPictureInfo³¹⁰, SetPictureInfo³⁶³, GetCurrentPictureInfo⁵¹³, SetCurrentPictureInfo⁵⁵⁹, SetPictureInfoEd⁵⁶⁵) now work with both pictures and *hot pictures*.

New properties for text and paragraph formatting:

- TFontInfo.StyleEx⁷⁰⁸ (new option – *rvfsAllCaps*);
- TFontInfo.CharScale⁶⁹⁹;
- TParaInfo.LineSpacing⁷²²;
- TParaInfo.LineSpacingType⁷²³;
- TParaInfo.Options⁷²³.

New feature: Tab navigation through links and inserted controls

TRichView.TabNavigation²⁵⁵ property;

TRichView.GetFocusedItem²⁹⁵ 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 reformatting²⁸⁸.

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

Miscellaneous

- OnCaretMove⁵⁷²
- GetLastCheckpoint³⁰⁵, GetPrevCheckpoint³¹¹
- two new possible values of Direction parameter of OnCaretGetOut⁵⁷¹
- TRVStyle.ParaStyles.FindStyleWithAlignment⁶⁸⁴ and FindSuchStyle⁶⁸⁵
- "whole word" option for TRichView.SearchText³⁴⁶ and TRichViewEdit.SearchText⁵⁴³ (thanks to Louis Kessler)

Improved:

- support of third party graphic classes¹⁰⁶⁸
- RTF export
- HTML export: SaveHTML³³⁶ now retains right and justify alignments.

2.20 New in version 1.4

Compatibility issues

- TRichView.OnControlAction⁽³⁷³⁾ 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⁽¹⁸⁹⁾.

New methods

TRichView:

- AddNLATag⁽²⁷⁰⁾, AddItem⁽²⁶⁹⁾, GetItem⁽²⁹⁶⁾, GetItemNo⁽³⁰¹⁾,
- GetJumpPointLocation⁽³⁰⁴⁾
- GetItemCoords⁽²⁹⁸⁾, GetItemClientCoords⁽²⁹⁸⁾

TRichViewEdit:

- InsertItem⁽⁵²²⁾,
- GetCurrentItem⁽⁵⁰⁸⁾, GetCurrentItemEx⁽⁵⁰⁹⁾
- SetItemTextEd⁽⁵⁶⁴⁾, SetCurrentItemText⁽⁵⁵⁷⁾, GetCurrentItemText⁽⁵¹¹⁾,
- CanChange⁽⁴⁹⁸⁾
- BeginItemModify⁽⁴⁹⁵⁾, EndItemModify⁽⁵⁰³⁾
- ResizeCurrentControl⁽⁵⁴³⁾, AdjustControlPlacement2⁽⁴⁸⁹⁾

TFontInfos:

- FindStyleWithCharset⁽⁶⁷⁹⁾

New properties

- TParaInfo.NextParaNo⁽⁷³⁰⁾
- TRVStyle.DefCodePage⁽⁶³⁵⁾
- TRichView.HScrollVisible⁽⁸¹⁶⁾
- TRichView.RVData⁽²⁴⁷⁾
- TRichView.InplaceEditor⁽⁸¹⁷⁾
- New option (rvoAutoSwitchLang) in TRichViewEdit.EditorOptions⁽⁴⁷⁵⁾ (thanks to Pavel Zhelty)
- New states in TRVTextDrawStates⁽¹⁰³⁰⁾

New events:

- TRichView.OnCopy⁽³⁷⁴⁾
- TRichView.OnHTMLSaveImage⁽³⁷⁵⁾
- TRichViewEdit.OnCaretGetOut⁽⁵⁷¹⁾
- New "hidden" parameters in TRVStyle.OnDrawStyleText⁽⁶⁷¹⁾ and OnDrawTextBack⁽⁶⁷³⁾

Cursors

Two new cursors. Ability to use system default hypertext cursor⁽¹⁴²⁾

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)

2.21 New in version 1.3

The main additions in version 1.3

- Undo/Redo⁽¹¹⁵⁾ (many topics, this is the link to the overview);
- Unicode support⁽¹³⁰⁾ (many topics, this is the link to the overview);
- Events for custom drawing⁽⁶⁶⁴⁾ (several topics; this is the link to the list)
- TRichView.RTFOptions⁽²⁴⁵⁾, CopyRTF⁽²⁸²⁾, SaveRTF⁽³⁴³⁾, SaveRTFToStream⁽³⁴⁴⁾
- TRVPrint.OnPagePrepaint event
- TFontInfo.Protection⁽⁷⁰⁵⁾ property
- TRVStyle.UseSound⁽⁶⁵⁶⁾ property
- TRichView.AddTextBlockNL, SetItemText⁽³⁶⁰⁾, GetItemText⁽³⁰³⁾, SaveTextToStream, SaveTextToStreamW⁽³⁴⁵⁾ methods;
- TRichView.OnControlAction⁽³⁷³⁾ event;
- new options in TRichView.Options⁽²⁴⁰⁾, TRichViewEdit.EditorOptions⁽⁴⁷⁵⁾;
- new parameter in TRichView.LoadText⁽³³⁰⁾;
- TFontInfo.IsEqual⁽⁷¹¹⁾, TFontInfos.FindStyleWithFontStyle⁽⁶⁸¹⁾, FindStyleWithFontSize⁽⁶⁸⁰⁾, FindStyleWithColor⁽⁶⁷⁹⁾, FindStyleWithFontName⁽⁶⁸⁰⁾, FindSuchStyle⁽⁶⁸¹⁾, FindStyleWithFont⁽⁶⁷⁹⁾
- new global variables⁽¹⁰⁴⁰⁾
- 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⁽⁷⁰⁵⁾.
- Older version of RichView will not be able to read RVF⁽¹²⁴⁾ with Unicode.

2.22 New in version 1.2

The main additions in version 1.2

- Borders and Background of Paragraphs⁽⁹⁷⁾;
- spacing between paragraphs (TParaInfo.SpaceBefore⁽⁷²⁶⁾, TParaInfo.SpaceAfter⁽⁷²⁶⁾) and ability to move items to the new line inside paragraphs (**Shift + Enter** in editor, also see TRichView.IsFromNewLine⁽³¹⁷⁾, TRichView.IsParaStart⁽³¹⁸⁾, TRichView.SetAddParagraphMode⁽³⁵⁰⁾);
- ability to specify explicit page breaks (TRVStyle.PageBreakColor⁽⁶⁴⁷⁾, TRichView.PageBreaksBeforeItems⁽²⁴⁴⁾, TRichView.Options⁽²⁴⁰⁾, TRichViewEdit.InsertPageBreak⁽⁵²⁵⁾, TRichViewEdit.RemoveCurrentPageBreak⁽⁵⁴²⁾);
- ability to change background color of text under mouse (TFontInfo.HoverBackColor⁽⁷⁰²⁾);
- ability to save and load styles to registry (TRVStyle.LoadReg⁽⁶⁶²⁾, TRVStyle.SaveReg⁽⁶⁶⁴⁾, Delphi 4+ is required);
- ability to search upwards in RichView (TRichView.SearchText⁽³⁴⁶⁾);
- ability to save background in RVF file or stream (optionally) (TRichView.RVFOptions⁽²⁴⁷⁾, RVF Overview⁽¹²⁴⁾).

Compatibility issues

- default search direction in TRichView.SearchText⁽³⁴⁶⁾ 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⁽⁶⁶³⁾);
- 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⁽⁷⁰²⁾

- TRVStyle.LoadReg ⁶⁶²
- TRVStyle.SaveReg ⁶⁶⁴

- TRVRect ⁹⁷⁵
- TRVBooleanRect ⁹⁶¹
- TRVBorderStyle ⁹⁹⁴

- Borders and Background of Paragraphs ⁹⁷

- TParaInfo.SpaceBefore ⁷²⁶
- TParaInfo.SpaceAfter ⁷²⁶
- TParaInfo.Border ⁷²¹
- TParaInfo.Background ⁷²¹

- TRVBorder ⁷⁴²
- TRVBorder Properties ⁷⁴³
- TRVBorder.BorderOffsets ⁷⁴³
- TRVBorder.Color ⁷⁴⁴
- TRVBorder.InternalWidth ⁷⁴⁴
- TRVBorder.Style ⁷⁴⁴
- TRVBorder.VisibleBorders ⁷⁴⁴
- TRVBorder.Width ⁷⁴⁴

- TRVBackgroundRect ⁷⁴¹
- TRVBackgroundRect Properties ⁷⁴²
- TRVBackgroundRect.BorderOffsets ⁷⁴²
- TRVBackgroundRect.Color ⁷⁴²

- TRichView.IsFromNewLine ³¹⁷
- TRichView.SetAddParagraphMode ³⁵⁰

- TRVStyle.PageBreakColor ⁶⁴⁷
- TRichView.PageBreaksBeforeItems ²⁴⁴
- TRichViewEdit.InsertPageBreak ⁵²⁵
- TRichViewEdit.RemoveCurrentPageBreak ⁵⁴²

- TRVStyle.SaveCSSToStream ⁶⁶³
- TRichView.SaveHTMLToStream ³⁴⁰
- TRichView.SaveHTMLToStreamEx ³⁴⁰

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

- RichView Paragraphs ⁹⁵, Building RichView Document ¹⁰²
- Export of RichView contents to HTML ¹²⁷, RichView Methods for Saving and Loading ¹²², Searching and Replacing ¹¹⁰,
- RVF Overview ¹²⁴, RVF Specification ¹⁴³
- TRVSearchOption ¹⁰²⁴
- TRichView.SearchText ³⁴⁶
- TRichViewEdit.SearchText ⁵⁴³

- TRichView.IsParaStart ⁽³¹⁸⁾
- TRichView.Options ⁽²⁴⁰⁾
- TRichView.RVFOptions ⁽²⁴⁷⁾
- TRichView.SaveHTML ⁽³³⁶⁾
- TRichView.SaveHTMLEx ⁽³³⁷⁾
- TRVSaveOption ⁽¹⁰²⁰⁾
- TRichView.SavePicture ⁽³⁴²⁾

Removed:

- TRVStyle.SaveCSSFile

3 Overview

Parts of Documents

- TRichView item types ⁽⁸⁵⁾
- TRichView items' "checkpoints" ⁽⁸⁷⁾
- TRichView items' "tags" ⁽⁹¹⁾
- TRichView hypertext overview ⁽⁹²⁾
- TRichView paragraphs ⁽⁹⁵⁾
- Borders and background of paragraphs ⁽⁹⁷⁾
- Styles and style templates ⁽⁹⁸⁾
- Hidden text ⁽¹⁰¹⁾

Basic Operations

- Building TRichView document ⁽¹⁰²⁾
- Scrolling in TRichView ⁽¹⁰⁵⁾
- Selecting part of TRichView document ⁽¹⁰⁷⁾
- TRichView Clipboard functions ⁽¹⁰⁸⁾
- Searching and replacing in TRichView and TRichViewEdit ⁽¹¹⁰⁾
- Obtaining items of RichView ⁽¹¹¹⁾
- Modifying TRichView items ⁽¹¹²⁾

Editing

- TRichViewEdit: Inserting items at position of caret ⁽¹¹⁴⁾
- Undo/redo in RichViewEdit ⁽¹¹⁵⁾
- Drag&drop ⁽¹¹⁸⁾

Graphics

- Animated images ⁽¹¹⁹⁾
- Smooth image scaling ⁽¹²⁰⁾
- Semitransparent objects ⁽¹²¹⁾

Saving and Loading

- TRichView methods for saving and loading ⁽¹²²⁾
- RichView Format (RVF) Overview ⁽¹²⁴⁾
- Export to HTML ⁽¹²⁷⁾

Internationalization

- Unicode⁽¹³⁰⁾
- Bi-directional text⁽¹³²⁾

Spelling Check

- Live spelling check⁽¹³²⁾

Technical Information

- Viewer vs editor⁽¹³⁵⁾
- Controls, documents, items⁽¹³⁶⁾
- Valid documents⁽¹³⁸⁾
- Units of measurement⁽¹³⁹⁾
- Custom drawing⁽¹⁴¹⁾
- Cursors⁽¹⁴²⁾
- RichView Format (RVF) specification⁽¹⁴³⁾
- Lazarus⁽¹⁵⁴⁾
- FireMonkey⁽¹⁵⁵⁾
- Skia and Skia4Delphi⁽¹⁵⁷⁾

3.1 Parts of documents

Overview⁽⁸⁴⁾ | Parts of Documents

- TRichView item types⁽⁸⁵⁾
- TRichView items' "checkpoints"⁽⁸⁷⁾
- TRichView items' "tags"⁽⁹¹⁾
- TRichView hypertext overview⁽⁹²⁾
- TRichView paragraphs⁽⁹⁵⁾
- Borders and background of paragraphs⁽⁹⁷⁾
- Styles and style templates⁽⁹⁸⁾
- Hidden text⁽¹⁰¹⁾

3.1.1 TRichView item types

Document in TRichView can contain items of the following types:

- **Text**⁽¹⁶¹⁾ can have different fonts. Each text item is linked to one item in the collection of text attributes ("styles")⁽⁶⁵⁴⁾. A text style defines a font of the text item and some additional effects (such as background color, vertical offset (for subscripts/superscripts), underline), 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⁽¹³⁰⁾ encoding.
- **Tabulators**⁽¹⁶²⁾ are used instead of TAB (#9) characters.
- **Pictures**⁽¹⁶³⁾: you can insert any Delphi graphic type in RichView. You can use TBitmap, TIcon, TMetafile, TjpegImage or third-party graphic classes⁽¹⁰⁶⁸⁾. Gif images can be animated.
- **"Hot Pictures"**⁽¹⁶⁶⁾: the same as pictures, but they are also hypertext links.

- **"Bullets"**⁽¹⁷⁰⁾ 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"**⁽¹⁷²⁾ are bullets-"hypertext links". They can specify two pictures: normal and "hot" (for highlighting effect under the mouse pointer)
- **"Breaks"**⁽¹⁶⁷⁾ are horizontal lines.
- **Controls**⁽¹⁶⁸⁾: You can use any Delphi/C++Builder control merged in text.
- **List Markers**⁽¹⁷⁴⁾ represent paragraph bullets & numbering.
- **Tables**⁽¹⁷³⁾ arrange subitems in rows and columns.
- **Labels**⁽¹⁷⁶⁾ are non-text items looking like a text.
- **Numbered sequences**⁽¹⁷⁷⁾ allow to insert numbered fields (like "Fig. 1", "Fig. 2", ...) in text.
- **Endnotes**⁽¹⁷⁸⁾ .
- **Footnotes**⁽¹⁸⁰⁾ .
- **Sidenotes**⁽¹⁸¹⁾ are notes displayed in floating boxes.
- **Text boxes**⁽¹⁸³⁾
- **References to parent note**⁽¹⁸⁴⁾
- **"Page number" fields**⁽¹⁸⁵⁾
- **"Page count" fields**⁽¹⁸⁶⁾
- **Mathematical expressions**⁽¹⁸⁷⁾ [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")⁽⁶⁴⁸⁾ 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⁽¹²⁴⁾). You can also export document as HTML, RTF, DocX file or print it (see export to html⁽¹²⁷⁾).

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⁽⁸⁴⁴⁾.

See also: detailed overview of item types⁽¹⁵⁸⁾

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⁽⁹¹⁾);
- *RaiseEvent* flag; if *RaiseEvent*=True then TRichView generates OnCheckpointVisible⁽³⁷²⁾ 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⁽¹²⁴⁾;
- 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⁽²⁴⁰⁾. 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⁽⁶³⁴⁾ for "normal" checkpoints (*clGreen* by default)
 - RichView.Style.CheckpointEvColor⁽⁶³⁴⁾ for checkpoints with *RaiseEvent*=True (*clLime* by default)
- You can create your own procedure for drawing checkpoints, see TRVStyle.OnDrawCheckpoint⁽⁶⁶⁷⁾.

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⁽²⁶⁴⁾ 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⁽³⁵³⁾ method, and remove it by RemoveCheckpoint⁽³³²⁾.

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⁽²⁹⁴⁾, GetNextCheckpoint⁽³⁰⁷⁾(CheckpointData), GetLastCheckpoint⁽³⁰⁵⁾, GetPrevCheckpoint⁽³¹¹⁾(CheckpointData) – allow you to obtain a list of all checkpoints in the document (except for checkpoints in table⁽¹⁹⁰⁾ cells, see below);
- GetItemCheckpoint⁽²⁹⁸⁾(ItemNo) – returns the checkpoint associated with the specified item;
- GetCheckpointByNo⁽²⁹¹⁾(No) – returns the checkpoint with the specified index;
- FindCheckpointByName⁽²⁸⁷⁾(Name) – returns the first checkpoint having the specified Name;
- FindCheckpointByTag⁽²⁸⁷⁾(Tag) – returns the first checkpoint having the specified Tag;
- event OnCheckpointVisible⁽³⁷²⁾ (Sender: TRichView; CheckpointData: TCheckpointData); this event is generated when:
 - RichView.CPEventKind⁽²²⁷⁾ = *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⁽²⁹¹⁾(CheckpointData, Tag, Name, RaiseEvent) – returns Tag, Name and "RaiseEvent" flag of checkpoint;
- GetCheckpointXY⁽²⁹²⁾(CheckpointData, X,Y), GetCheckpointYEx⁽²⁹³⁾(CheckpointData): TRVCoord⁽⁹⁹⁸⁾ – obtaining information about coordinates of checkpoint (Y is the most useful; you can pass this Y to RichView.ScrollTo⁽⁸²⁰⁾ method);
- GetCheckpointItemNo⁽²⁹¹⁾(CheckpointData) : Integer – returns the index of the associated item; it can return -1 for checkpoint in the end of document (no associated item);
- GetCheckpointNo⁽²⁹²⁾(CheckpointData): Integer – returns the index of the checkpoint;

Methods belonging to the both groups:

- GetCheckpointY⁽²⁹²⁾(no): TRVCoord⁽⁹⁹⁸⁾ – equivalent to GetCheckpointYEx(GetCheckpointByNo(No));

Methods of editor

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

- SetCheckpointInfoEd⁽⁵⁴⁹⁾ – adds checkpoint for the specified item, as an editing operation;
- RemoveCheckpointEd⁽⁵⁴¹⁾ – 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⁽⁵⁰⁶⁾ – returns checkpoint associated with the item at the caret position;

- `SetCurrentCheckpointInfo`⁽⁵⁵³⁾ – adds/modifies checkpoint for the item at the caret position, as an editing operation;
- `RemoveCurrentCheckpoint`⁽⁵⁴¹⁾ – 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`⁽⁵¹⁶⁾ – 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`⁽⁵⁰³⁾ – returns the checkpoint exactly at the caret position;
- `RemoveCheckpointAtCaret`⁽⁵⁴¹⁾ – deletes the checkpoint exactly at the caret position, as an editing operation.

RTF, DocX, and HTML

In HTML⁽¹²⁷⁾, checkpoints are saved⁽³³⁵⁾ as anchors (<a name> tag). Either checkpoint name or checkpoint index can be used, see `HTMLSaveProperties`⁽²³⁷⁾.`UseCheckpointNames`⁽⁴³⁷⁾. Links to checkpoints must start with '#' character. When importing HTML, anchors are imported as checkpoints. Optionally, "id" attributes of tags can be imported as checkpoints, see `HTMLReadProperties`⁽²³⁷⁾.`IDAsCheckpoints`⁽⁴²⁷⁾.

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(1029);
    Name: TRVUnicodeString(1032);
    RaiseEvent: Boolean;
    CheckpointData: TCheckpointData;
begin
    CheckpointData := MyRichView.GetFirstCheckPoint(294);
    while CheckpointData <> nil do begin
        MyRichView.GetCheckpointInfo(291)(CheckpointData, Tag, Name, RaiseEvent);
        // processing this checkpoint
        ...
        CheckpointData := MyRichView.GetNextCheckpoint(307)(CheckpointData);
    end;
end;
```

Example 2

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

```
procedure EnumCheckpoints(RVData: TCustomRVData(957));
var i, r, c: Integer;
    table: TRVTableItemInfo(846);
```

```

Tag: TRVTag1029;
Name: TRVUnicodeString1032;
RaiseEvent: Boolean;
CheckpointData: TCheckpointData;
begin
  for i := 0 to RVDData.ItemsCount-1 do begin
    CheckpointData := RVDData.GetItemCheckpoint298(i);
    if CheckpointData<>nil then begin
      RVDData.GetCheckpointInfo291(CheckpointData, Tag, Name, RaiseEvent);
      // processing this checkpoint
      ...
    end;
    if RVDData.GetItem296(i) is TRVTableItemInfo846 then
      begin
        table := TRVTableItemInfo846(RVDData.GetItem296(i));
        for r := 0 to table.RowCount864-1 do
          for c := 0 to table.ColCount858-1 do
            if table.Cells857[r,c]<>nil then
              EnumCheckpoints(table.Cells857[r,c].GetRVData907);
          end;
        end;
      end;
end;
end;
Call:
EnumCheckpoints(MyRichView.RVDData247);

```

In the second example, `RVDData.GetCheckpointXY292` returns checkpoint coordinates relative to the top left corner of this `RVDData` (it may be left top corner of the cell). In order to get absolute coordinates of this checkpoint in the document (for `MyRichView.ScrollTo820`), obtain the coordinates of top left corner of this `RVDData` (`TCustomRVFormattedData(RVDData).GetOriginEx(X,Y)`), relative coordinates of checkpoint (`TCustomRVFormattedData(RVDData).GetCheckpointXY(X,Y)`), and sum them up

See also...

See also properties and events of TRVStyle:

CheckpointColor, CheckpointEvColor⁶³⁴, OnDrawCheckpoint⁶⁶⁷

See also properties of TRichView:

Options²⁴⁰, CPEventKind²²⁷

See also events of TRichView:

OnCheckpointVisible³⁷²

See also methods of TRichView:

AddCheckpoint²⁶⁴, SetCheckpointInfo³⁵³, RemoveCheckpoint³³², GetFirstCheckpoint²⁹⁴,
 GetNextCheckpoint³⁰⁷, GetItemCheckpoint²⁹⁸, GetCheckpointByNo²⁹¹, FindCheckpointByName²⁸⁷,
 FindCheckpointByTag²⁸⁷, GetCheckpointInfo²⁹¹, GetCheckpointXY²⁹², GetCheckpointYEx²⁹³,
 GetCheckpointItemNo²⁹¹, GetCheckpointNo²⁹², GetCheckpointY²⁹²

See also methods of **TRichViewEdit**:

GetCurrentCheckpoint⁽⁵⁰⁶⁾, SetCheckpointInfoEd⁽⁵⁴⁹⁾, RemoveCheckpointEd⁽⁵⁴¹⁾

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"⁽⁸⁷⁾ also have tags; all rules for items tags work for checkpoints tags;
- table cells also have tags⁽⁹⁰⁶⁾; 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⁽⁴⁷⁵⁾;
- 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⁽¹⁰²⁾ which do not have Tag parameter. These functions create new items with Tag="".

Unicode note: tags are Unicode strings.

Example:

```
var Tag: TRVTag(1029);
    S: TRVUnicodeString(1032);
// Converting the first text item and its tag string to upper case
MyRichView.GetTextInfo(313)(0, S, Tag);
MyRichView.SetItemText(360)(0, UpperCase(S));
MyRichView.SetItemTag(360)(0, UpperCase(Tag));
```

Tags in older version of TRichView

In TRichView versions prior to v13.2, tags were Integers. Depending on *rvoTagsArePChars* Option⁽²⁴⁰⁾, 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⁽²⁷⁰⁾, AddBreak⁽²⁶²⁾, AddPicture⁽²⁷²⁾, AddHotspot⁽²⁶⁸⁾, AddBullet⁽²⁶³⁾, AddControl⁽²⁶⁵⁾, AddBreak⁽²⁶²⁾

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

AddCheckpoint⁽²⁶⁴⁾.

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

GetItemTag⁽³⁰²⁾, GetBreakInfo⁽²⁸⁹⁾, GetBulletInfo⁽²⁸⁹⁾, GetHotspotInfo⁽²⁹⁵⁾, GetPictureInfo⁽³¹⁰⁾, GetControlInfo⁽²⁹³⁾, GetTextInfo⁽³¹³⁾.

TRichView methods for modifying items (including tags):

SetItemTag⁽³⁶⁰⁾, SetBreakInfo⁽³⁵¹⁾, SetBulletInfo⁽³⁵²⁾, SetHotspotInfo⁽³⁵⁷⁾, SetPictureInfo⁽³⁶³⁾, SetControlInfo⁽³⁵⁴⁾.

TRichView methods allowing work with checkpoints tags:

FindCheckpointByTag⁽²⁸⁷⁾, GetCheckpointInfo⁽²⁹¹⁾, SetCheckpointInfo⁽³⁵³⁾.

TRichViewEdit methods for obtaining information about item in caret position:

GetCurrentTag⁽⁵¹⁴⁾, GetCurrentBreakInfo⁽⁵⁰⁴⁾, GetCurrentBulletInfo⁽⁵⁰⁵⁾, GetCurrentHotspotInfo⁽⁵⁰⁷⁾, GetCurrentPictureInfo⁽⁵¹³⁾, GetCurrentControlInfo⁽⁵⁰⁶⁾, GetCurrentTextInfo⁽⁵¹⁴⁾.

TRichViewEdit methods for inserting:

InsertStringTag, InsertStringATag, InsertStringWTag⁽⁵³⁰⁾

TRichViewEdit methods for modifying items (including tags):

SetItemTagEd⁽⁵⁶³⁾, SetBreakInfoEd⁽⁵⁴⁷⁾, SetBulletInfoEd⁽⁵⁴⁸⁾, SetHotspotInfoEd⁽⁵⁶⁰⁾, SetPictureInfoEd⁽⁵⁶⁵⁾, SetControlInfoEd⁽⁵⁵⁰⁾.

TRichViewEdit methods for modifying information about item in caret position:

SetCurrentTag⁽⁵⁶⁰⁾, SetCurrentBreakInfo⁽⁵⁵¹⁾, SetCurrentBulletInfo⁽⁵⁵²⁾, SetCurrentHotspotInfo⁽⁵⁵⁵⁾, SetCurrentPictureInfo⁽⁵⁵⁹⁾, SetCurrentControlInfo⁽⁵⁵⁴⁾

TRichViewEdit methods for working with checkpoints tags:

SetCheckpointInfoEd⁽⁵⁴⁹⁾, SetCurrentCheckpointInfo⁽⁵⁵³⁾.

TRichViewEdit options affecting tags:

EditorOptions⁽⁴⁷⁵⁾.

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⁽⁷¹⁴⁾;
- tabs⁽¹⁶²⁾ and label items⁽¹⁷⁶⁾, linked to hypertext styles⁽⁷¹⁴⁾;
- *hot pictures*⁽¹⁶⁶⁾
- *hotspots*⁽¹⁷²⁾

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

- changing color of text items (TFontInfo HoverColor⁽⁷⁰²⁾, HoverBackColor⁽⁷⁰²⁾);
- underlining text items (TFontInfo HoverEffects⁽⁷⁰³⁾);

- changing mouse cursor (any hypertext text style can have its own cursor, see `TRVStyle.JumpCursor`⁽⁶⁴³⁾, `TFontInfo.JumpCursor`⁽⁷⁰⁴⁾)
- for hotspots⁽¹⁷²⁾ 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`⁽³⁹⁵⁾, `OnJump`⁽³⁸²⁾) and cursor indication do not work in editor (`TRichViewEdit`⁽⁴⁶¹⁾). 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`⁽⁴⁷⁵⁾.

Since version 1.6, hypertext works in `ReadOnly`⁽⁴⁸⁰⁾ editor just like in viewer (since version 13, this behavior can be turned off using `rvoNoReadOnlyJumps` option from `EditorOptions`⁽⁴⁷⁵⁾)

Besides, you can provide some specific actions in response on user clicking any RichView item, using `OnRVMouseDown`⁽³⁹⁴⁾ and `OnRVMouseUp`⁽³⁹⁶⁾ events.

Hypertext

Any hypertext link (text, *hotspot* or *hot picture*) has its own **id**. This **id** is passed in hypertext events - `OnRVMouseMove`⁽³⁹⁵⁾ and `OnJump`⁽³⁸²⁾. The first hypertext item has **id** equal to `RichView.FirstJumpNo`⁽²³⁶⁾, 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`⁽³⁹⁵⁾ 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`⁽³⁸²⁾.

▼ 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`⁽³⁰⁴⁾. 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 ..., CRVFDData, ShellApi;
...
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var URL: String;
    RVDData: TCustomRVFormattedData(957);
    ItemNo: Integer;
begin
    MyRichView.GetJumpPointLocation(304)(id, RVDData, ItemNo);
    URL := RVDData.GetItemText(303)(ItemNo);
    ShellExecute(Application.Handle, 'open', PChar(URL),
        nil, nil, SW_NORMAL);
end;
```

In the example above, we use `GetJumpPointLocation`⁽³⁰⁴⁾ to get the document containing this hyperlink (RVDData) 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 `RVDData.GetItemStyle`⁽³⁰²⁾ method to recognize the item type ("style"), and next use `GetTextInfo`⁽³¹³⁾ or `GetHotspotInfo`⁽²⁹⁵⁾/`GetPictureInfo`⁽³¹⁰⁾. 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`⁽⁹¹⁾).

▼ Example

```
uses ..., CRVFDData, ShellApi;
...
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var URL: String;
    RVDData: TCustomRVFormattedData(957);
    ItemNo: Integer;
begin
    MyRichView.GetJumpPointLocation(304)(id, RVDData, ItemNo);
    URL := RVDData.GetItemTag(302)(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.GetJumpPointLocation304(id: Integer;
  var RVDData: TCustomRVFormattedData; var ItemNo: Integer);
```

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

RVDData can be RichView[Edit].RVDData²⁴⁷, cell⁸⁹⁵, or RVDData²⁴⁷ of inplace editor.

See also...

- OnReadHyperlink³⁸⁸, OnWriteHyperlink⁴¹¹;
- 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.Style²⁵³.ParaStyles⁶⁴⁸).

For the list of properties available for paragraphs, see TParaInfo⁷¹⁸ class (TParaInfo is a class of item in the collection of paragraph styles).

Line breaks inside paragraphs

(Since version 1.2) You can move an 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:

- FirstIndent⁷²² is not applied to the first line after the break;
- no additional spacing between this line and the previous line (SpaceBefore⁷²⁶ and SpaceAfter⁷²⁶) is applied;
- if the paragraph has border/background⁹⁷, 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 breaks¹⁶⁷ or tables⁸⁴⁶*).

Breaks (horizontal lines) do not have paragraph style, so AddBreak method does not have ParaNo parameter. But SpaceBefore⁷²⁶ and SpaceAfter⁷²⁶ values of the 0-th paragraph style are added to spacing of all breaks.

These methods are:

```
procedure AddNL270(s, StyleNo, ParaNo, Tag);
procedure AddFmt266(FormatStr, Args, StyleNo, ParaNo);
```

```

procedure AddTextNL(274)(s, StyleNo, FirstParaNo, OtherParaNo,
  DefAsSingleParagraph, Tag);
procedure AddPicture(272)(Name, gr, ParaNo, VAlign, Tag);
procedure AddHotspot(268)(Name, ImageIndex, HotImageIndex,
  ImageList, ParaNo, VAlign, Tag);
procedure AddBullet(263)(Name, ImageIndex, ImageList, ParaNo, VAlign, Tag);
procedure AddControl(265)(Name, ctrl, ParaNo, VAlign, Tag);
procedure Add(270)(s, StyleNo); // equivalent to AddNL(s, StyleNo, -1).

```

See also: building a document⁽¹⁰²⁾, item types⁽⁸⁵⁾.

See also methods of TRichView:

```

function LoadText(330)(FileName, StyleNo, ParaNo, AsSingleParagraph,
  CodePage):Boolean;
function LoadTextW(330)(FileName, StyleNo, ParaNo,
  DefAsSingleParagraph):Boolean;
function AppendRVFFFromStream(276)(Stream, ParaNo):Boolean;
function IsParaStart(318)(ItemNo): Boolean;
function IsFromNewLine(317)(ItemNo): Boolean;
function GetItemPara(301)(ItemNo): Integer;

```

Text flow around left- and right-aligned items

Some items (for example, pictures⁽¹⁶³⁾) can be aligned to the left or the right side (see TRVVAlign⁽¹⁰³³⁾). 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⁽²²⁵⁾ and/or ClearRight⁽²²⁶⁾ 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(473): Integer;

```

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

When the user moves the caret to paragraph with different style, OnCurParaStyleChanged⁽⁵⁷³⁾ 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(490)(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(491)(UserData: Integer);

```

See also...

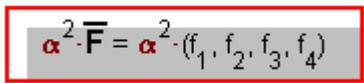
Borders and Background of Paragraphs ⁽⁹⁷⁾

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 ⁽⁷¹⁸⁾). Border is defined in Border ⁽⁷²¹⁾ property, background is in Background ⁽⁷²¹⁾ property.

By default, there is no border around paragraph (TRVStyle.ParaStyles[i].Border.Style ⁽⁷⁴⁴⁾=rvbsNone), and background of paragraph is transparent (TRVStyle.ParaStyles[i].Background.Color=c/None).

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.

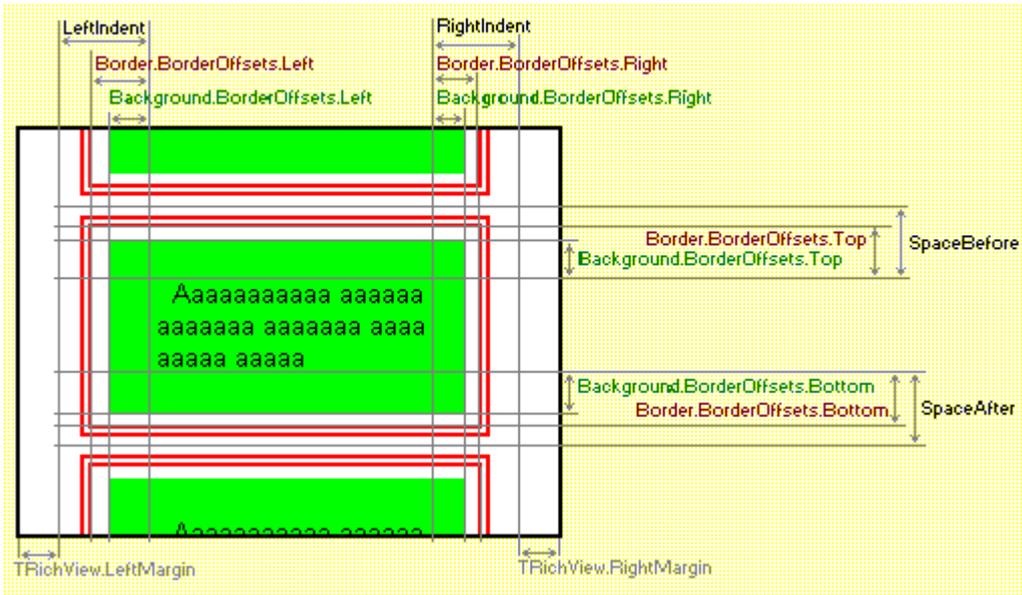
Top, right, bottom, left offsets (i.e. gap and padding) are defined by Top, Right, Bottom, Left properties of BorderOffsets.

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 ⁽⁷²⁶⁾ property of the same paragraph.

BorderOffsets.Bottom must not exceed SpaceAfter ⁽⁷²⁶⁾ property of the same paragraph.



Scheme: properties of paragraph border, background and layout

You can hide some edges of border using `Border.VisibleBorders`⁽⁷⁴⁴⁾ property.

`Breaks`⁽¹⁶⁷⁾ 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`⁽⁷²¹⁾

`TRVBorder`⁽⁷⁴²⁾

3.1.7 Styles and styleTemplates

As you could see, this manual sometimes calls items in `TextStyles`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾ "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`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾, and they are not changed while the application works.

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

▼ Example

In this example, `RichView1: TRichView`⁽²¹⁰⁾ is linked to `RVStyle1: TRVStyle`⁽⁶³⁰⁾.

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

```

    StyleName(696) := 'Normal Text';
end;
with RVStyle1.TextStyles.Add do begin
    StyleName := 'Title Text';
    Size(707) := 14;
    Style(708) := [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(719) := rvaCenter;
    Options(723) := [rvpaoKeepWithNext];
end;
...
RichView1.Clear(279);
RichView1.AddNL(270)('Chapter 1', 1, 1);
RichView1.AddNL('Once upon a time...', 1, 1);
RichView1.Format(288);

```

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(701) := clRed;
    FontName(701) := 'Verdana';
end;
with RVStyle1.ParaStyles[1] do begin
    Alignment := rvaLeft;
end;
RichView1.Reformat(332);

```

You can configure⁽²¹⁸⁾ 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⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾; 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⁽⁴⁹²⁾, ApplyParaStyleConversion⁽⁴⁹¹⁾). RichViewActions add new items as well.

In this mode, items of TextStyles⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾ 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⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾ may be marked as Standard⁽⁶⁹⁵⁾ (usually, the items added initially). They may have a special meaning, they are not removed by DeleteUnusedStyles⁽²⁸⁶⁾,

they can be saved in RTF as a style sheet (if `rvrtfSaveStyleSheet` in `RTFOptions`⁽²⁴⁵⁾). 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`⁽⁶⁵²⁾ is a collection in `TRVStyle`⁽⁶³⁰⁾ component. Unlike `TextStyles`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾, style templates do not define text and paragraph properties of document items⁽¹⁵⁸⁾ and paragraphs directly. Instead, they provide a mechanism for modifying `TextStyles`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾.

Using style templates is optional; they are used only if `TRichView.UseStyleTemplates`⁽²⁵⁶⁾ = `True`.

Document items do not have direct links to style templates, they are still linked with `TextStyles`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾. But items in `TextStyles`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾ may be linked to style templates using `StyleTemplateId`⁽⁶⁹⁶⁾ 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⁽¹⁶¹⁾, tab⁽¹⁶²⁾, label⁽¹⁷⁶⁾, sequence⁽¹⁷⁷⁾, footnote⁽¹⁸⁰⁾, endnote⁽¹⁷⁸⁾, or note reference⁽¹⁸⁴⁾).

It's recommended (but not necessary) to link every paragraph style to some style template. The recommended name⁽⁷³⁵⁾ 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⁽⁷³⁵⁾ of style templates: applicable to text, applicable to paragraphs, applicable to text and paragraphs.

Each style template may have properties of both text⁽⁷³⁷⁾ and paragraph⁽⁷³⁷⁾. Unlike items of `TextStyles` or `ParaStyles`, a style template:

- may define only a subset of text and paragraph properties (listed in `ValidTextProperties`⁽⁷³⁸⁾ and `ValidParaProperties`⁽⁷³⁸⁾);
- may inherit text and paragraph properties from a parent style template⁽⁷³⁷⁾.

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`⁽⁶⁵⁴⁾[`StyleNo`].`StyleTemplateId`⁽⁶⁹⁶⁾, a paragraph style template as `TextStyles`⁽⁶⁵⁴⁾[`StyleNo`].`ParaStyleTemplateId`⁽⁷⁰⁵⁾. `TRichView` automatically provides for each document item that `TextStyles`⁽⁶⁵⁴⁾[`StyleNo`].`ParaStyleTemplateId`⁽⁷⁰⁵⁾ = `ParaStyles`⁽⁶⁴⁸⁾[`ParaNo`].`StyleTemplateId`⁽⁶⁹⁶⁾. 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`⁽⁶⁵⁴⁾ and `ParaStyles`⁽⁶⁴⁸⁾ 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`⁽⁶⁵²⁾ are edited.

TRichViewEdit has the following methods for applying the specified style template to the selection: `ApplyStyleTemplate`⁽⁴⁹³⁾, `ApplyTextStyleTemplate`⁽⁴⁹⁴⁾, `ApplyParaStyleTemplate`⁽⁴⁹¹⁾. The same methods may be used to clear a style template or additional formatting.

Style templates can be edited using `ChangeStyleTemplates`⁽⁵⁰⁰⁾ 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`⁽²⁴⁰⁾ property of `TRichView`⁽²¹⁰⁾ control (they are displayed with a special dotted underline).

Text items⁽¹⁶¹⁾ are hidden, if `rvteoHidden` is included in the `Options`⁽⁷⁰⁴⁾ property of their style⁽⁷¹²⁾.

Non-text items⁽⁸⁵⁾ are hidden, if a non-zero value is assigned to `rvepHidden` extra item integer property⁽¹⁰⁰⁰⁾.

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`⁽²⁴⁶⁾.`SkipHiddenText`⁽⁴⁵⁸⁾ property of `TRichView`⁽²¹⁰⁾ control. By default (`False`), hidden text and objects are not loaded from RTF/DocX (skipped).

HTML: The CSS version of HTML export (`HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = `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`⁽²³⁷⁾.`SkipHiddenText`⁽⁴²⁸⁾ property of `TRichView`⁽²¹⁰⁾ 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 if `TRichView.MarkdownProperties.SaveHiddenText`⁽⁴⁴⁸⁾ = `True`.

Performance

A text formatting procedure is somewhat faster, if the option for displaying hidden items (`rvoShowHiddenText` in `Options`⁽²⁴⁰⁾) is set.

3.2 Basic Operations

Overview⁽⁸⁴⁾ | Basic Operations

- Building `TRichView` document⁽¹⁰²⁾
- Scrolling in `TRichView`⁽¹⁰⁵⁾
- Selecting part of `TRichView` document⁽¹⁰⁷⁾
- `TRichView` Clipboard functions⁽¹⁰⁸⁾
- Searching and replacing in `TRichView` and `TRichViewEdit`⁽¹¹⁰⁾

- Obtaining items of RichView⁽¹¹¹⁾
- Modifying TRichView items⁽¹¹²⁾.

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(279);
<Methods for adding contents>
MyRichView.Format(288);
```

Note: all methods introduced in RichView (except for the formatting methods) do not reformat document, do not redraw RichView and do not call OnChange⁽⁵⁷²⁾ 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⁽¹⁶¹⁾

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)⁽²⁷⁰⁾ – adds one item with text *S*; *StyleNo* is an index in the collection of text styles (TRichView.Style.TextStyles⁽⁶⁵⁴⁾); *ParaNo* is an index in the collection of paragraph styles (RichView.Style.ParaStyles⁽⁶⁴⁸⁾) (if this item will start a new paragraph) or -1 (if this item will continue the paragraph). Optionally, you can specify Tag⁽⁹¹⁾.
- Add(S, StyleNo)⁽²⁷⁰⁾ – equivalent to AddNL(S, StyleNo, -1).
- AddTextNL(s, StyleNo, FirstParaNo, OtherParaNo, Tag)⁽²⁷⁴⁾ – 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⁽⁹¹⁾.
- AddFmt(FormatStr, Args, StyleNo, ParaNo)⁽²⁶⁶⁾ adds Format(FormatStr, Args) string, where Format is a function from SysUtils unit.

Methods working with ANSI/Unicode explicitly

- AddNLA, AddNLW⁽²⁷⁰⁾, AddTextNLA, AddTextNLW⁽²⁷⁴⁾

Appending Images⁽¹⁶³⁾

You can add picture of any Delphi format – TBitmap, TIcon, TMetafile, TjpegImage or third party graphic format⁽¹⁰⁶⁸⁾.

You should create a picture and TRichView will destroy it itself when necessary (for example, when you call RichView.Clear⁽²⁷⁹⁾).

AddPicture(Name, gr, ParaNo, VAlign, Tag)⁽²⁷²⁾, where:

- Name: TRVUnicodeString⁽¹⁰³²⁾ – name of picture; this string is not used by TRichView itself; this string should not contain cr and lf characters;
- gr: TRVGraphic⁽⁹⁷⁰⁾ - 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⁽⁹¹⁾.

Appending Hot Images⁽¹⁶⁶⁾

- AddHotPicture⁽²⁶⁷⁾

Appending Delphi Controls⁽¹⁶⁸⁾

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)⁽²⁶⁵⁾ 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⁽¹⁷⁰⁾ – Images from ImageLists

AddBullet (Name, ImageIndex, ImageList, ParaNo, VAlign, Tag)⁽²⁶³⁾, where:

- Name: TRVUnicodeString⁽¹⁰³²⁾ – 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⁽⁹¹⁾.

Appending Hotspots⁽¹⁷²⁾ – 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)⁽²⁶⁸⁾ 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⁽¹⁶⁷⁾ – Horizontal Lines

- AddBreak.⁽²⁶²⁾

Adding Paragraphs' Bullets and Numbering⁽¹⁷⁴⁾ (list markers)

List markers can be added using SetListMarkerInfo⁽³⁶²⁾ 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⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, numbered sequences⁽¹⁷⁷⁾, endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾, references to parent footnotes and endnotes⁽¹⁸⁴⁾, page numbers⁽¹⁸⁵⁾, page counts⁽¹⁸⁶⁾, equations⁽¹⁸⁷⁾, custom item types⁽¹⁸⁹⁾ do not have special methods for adding them to documents. All of them are added by AddItem⁽²⁶⁹⁾ method.

Appending Checkpoints

Checkpoints are not items of TRichView, but the method for adding checkpoints is similar to methods for adding items: AddCheckpoint⁽²⁶⁴⁾. See "checkpoints"⁽⁸⁷⁾.

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(270) ('First page', 0, 0);
ItemNo := MyRichView.ItemCount(237);
MyRichView.AddNL(270) ('Second page', 0, 0);
MyRichView.PageBreaksBeforeItems(244)[ItemNo] := True;
```

Appending and Inserting RVF⁽¹²⁴⁾

You can append RVF data by the methods:

- function InsertRVFFromStream⁽³¹⁶⁾(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⁽²⁷⁶⁾(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`⁽⁶⁴⁸⁾) (if item will start a paragraph) or -1 (if the item will continue paragraph).

Appending Content of Another TRichView

- procedure `AppendFrom`⁽²⁷⁵⁾ (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⁽¹²²⁾ (**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`⁽²⁷⁴⁾), but there is a method that affects their behavior:

`TRichView.SetAddParagraphMode`⁽³⁵⁰⁾.

Some custom item types (such as tables⁽¹⁹⁰⁾) do not have special methods for appending them. They can be appended by general method `AddItem`⁽²⁶⁹⁾.

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`⁽²⁵⁷⁾ 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`⁽²⁸⁸⁾ method).

You can get or set position of vertical scrollbar using `VScrollPos`⁽⁸¹⁹⁾ property (in range `0..VScrollMax`⁽⁸¹⁸⁾)

You can also use method `ScrollTo`⁽⁸²⁰⁾ 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`⁽⁸¹⁹⁾ property.

DocumentHeight⁽²³³⁾ 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⁽²⁵⁵⁾ and BottomMargin⁽²²⁵⁾.

You can get vertical coordinate in document for:

- checkpoints⁽⁸⁷⁾
- hypertext link⁽⁹²⁾ (GetJumpPointY⁽³⁰⁵⁾)
- any item (GetItemCoords⁽²⁹⁸⁾)

Vertical scrolling [FireMonkey]

Vertical scrollbar appears automatically when necessary.

You can get or set position of vertical scrollbar using VScrollPos⁽⁸¹⁹⁾ property (in range 0..VScrollMax⁽⁸¹⁸⁾)

You can also use method ScrollToPosition⁽⁸²⁰⁾ for scrolling to the specified vertical position, measured in pixels.

DocumentHeight⁽²³³⁾ 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⁽²⁵⁵⁾ and BottomMargin⁽²²⁵⁾.

You can get vertical coordinate in document for:

- checkpoints⁽⁸⁷⁾
- hypertext link⁽⁹²⁾ (GetJumpPointY⁽³⁰⁵⁾)
- any item (GetItemCoords⁽²⁹⁸⁾)

Horizontal scrolling

Horizontal scrollbar appears automatically when necessary.

In VCL and LCL versions, you can hide it with HScrollVisible⁽⁸¹⁶⁾ property

You can get or set the position of horizontal scrollbar using HScrollPos⁽⁸¹⁶⁾ property (in range 0..HScrollMax⁽⁸¹⁶⁾)

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⁽²³⁸⁾, RightMargin⁽²⁴⁴⁾)

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

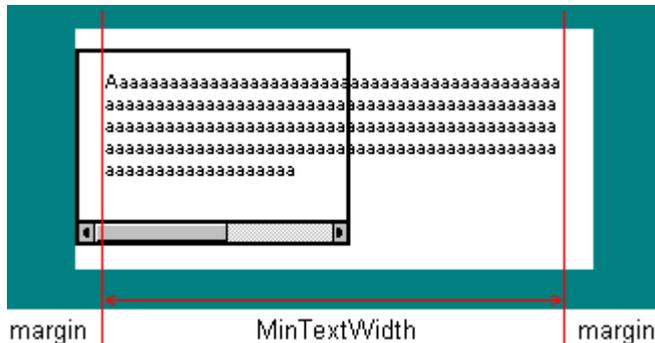
```
max(client width of TRichView,
    "maximal width of pictures or controls" +
    "maximal left(722) + first line(722) + right(725) indent" + margins),
```

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

Text in paragraphs with no-wrap option⁽⁷²³⁾ can also affect to width of scrolling area, in the same was as wide pictures or controls.

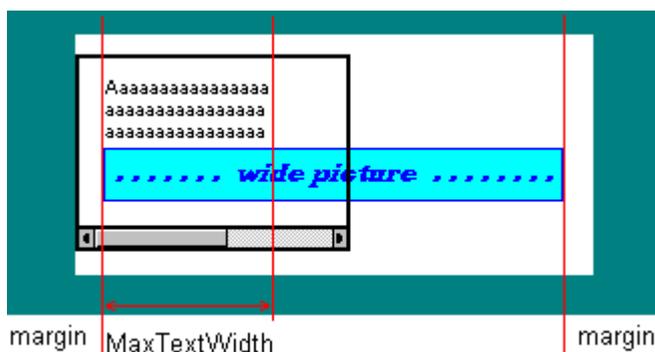
Use MinTextWidth⁽²⁴⁰⁾ property to set the minimal width for text wrapping (text wraps when its length in pixels exceeds this value). Horizontal scrollable area cannot be less than

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⁽²³⁹⁾ 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⁽²⁴⁰⁾. 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⁽⁷²³⁾ 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⁽⁶⁴⁹⁾.

Selection Appearance

Selected text has RichView.Style.SelTextColor/InactiveSelTextColor⁽⁶⁴¹⁾ color and RichView.Style.SelColor/InactiveSelColor⁽⁶⁴¹⁾ background.

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

See also [RVStyle.SelectionStyle](#) ⁽⁶⁴⁹⁾.

Operations

You can:

- disallow making selection (clear *rvoAllowSelection* in [Options](#) ⁽²⁴⁰⁾); please do not use in editor!
- determine if the selection is not empty ([SelectionExists](#) ⁽³⁴⁹⁾);
- deselect TRichView ([Deselect](#) ⁽²⁸⁶⁾; [Invalidate](#));
- select the whole document ([SelectAll](#) ⁽³⁴⁸⁾; [Invalidate](#));
- get selection bounds ([GetSelectionBounds](#) ⁽³¹²⁾);
- select a part of the document ([SetSelectionBounds](#) ⁽³⁶⁵⁾; [Invalidate](#));
- select the inserted control ([SelectControl](#) ⁽³⁴⁸⁾);
- get the selection as a text ([GetSelText](#), [GetSelTextW](#) ⁽³¹³⁾);
- get the selection as a Rich Text Format ([SaveRTFtoStream](#) ⁽³⁴⁴⁾, [SaveRTF](#) ⁽³⁴³⁾);
- get the selection as an image, if image is selected ([GetSelectedImage](#) ⁽³¹²⁾);
- save the selection to stream or file ([SaveRVFtoStream](#) ⁽³⁴⁴⁾, [SaveRVF](#) ⁽³⁴⁴⁾);
- copy the selection to the Clipboard in various formats (see [RichView Clipboard Functions Overview](#) ⁽¹⁰⁸⁾)

In editor you can:

- delete selection ([DeleteSelection](#) ⁽⁵⁰³⁾);
- select the current word ⁽⁵⁴⁵⁾ or line ⁽⁵⁴⁵⁾;
- cut selection to the Clipboard ([CutDef](#) ⁽⁵⁰²⁾);

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](#) ⁽¹¹⁰⁾

3.2.4 Clipboard

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

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

Copying [VCL, LCL, FMX for Windows and macOS]

The main method for copying to the Clipboard is [CopyDef](#) ⁽²⁸¹⁾. It can copy selection in one or more formats.

This method copies data in formats specified in [RichView.Options](#) ⁽²⁴⁰⁾:

Option	Meaning
<i>rvoAutoCopyText</i>	if set, CopyDef copies selection as ANSI text

Option	Meaning
<i>rvoAutoCopyImage</i>	if set, CopyDef can copy selected image to clipboard
<i>rvoAutoCopyUnicodeText</i>	if set, CopyDef copies selection as Unicode text
<i>rvoAutoCopyRTF</i>	if set, CopyDef copies selection as RTF (Rich Text Format)
<i>rvoAutoCopyRVF</i>	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⁽²⁸¹⁾ – copies selection to the Clipboard in all formats;
- CopyTextA⁽²⁸³⁾ – copies the selection as ANSI text;
- CopyTextW⁽²⁸³⁾ – copies selection as Unicode text
- CopyImage⁽²⁸¹⁾ – if there is a selected image, this method copies it to the Clipboard;
- CopyRVF⁽²⁸²⁾ – copies selection in RVF format⁽¹²⁴⁾. 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⁽¹²⁴⁾, RVF Specification⁽¹⁴³⁾)
- CopyRTF⁽²⁸²⁾ – copies selection as RTF (Rich Text Format).

SelectionExists⁽³⁴⁹⁾ method answers to the question "does the selection contain some data (not empty)?"

OnCopy⁽³⁷⁴⁾ event allows copying in your own formats.

Copying [FMX for non-Windows and non-macOS]

In FireMonkey (except for Windows and macOS platforms), only one format can be copied to the Clipboard. Copy⁽²⁸¹⁾ and CopyDef⁽²⁸¹⁾ work identically: they copy the most appropriate format from the formats listed in *rvoAutoCopy** Options⁽²⁴⁰⁾.

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

Pasting

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

Pasting:

- PasteBitmap⁽⁵³⁵⁾ – pastes bitmap from the Clipboard, if available;
- PasteMetafile⁽⁵³⁷⁾ – pastes metafile picture from the Clipboard, if available [VCL];
- PasteGraphicFile⁽⁵³⁵⁾ – pastes graphic file(s) the Clipboard, if available [Windows];
- PasteText⁽⁵³⁸⁾ – pastes text from the Clipboard, if available [VCL, LCL, FMX];
- PasteRVF⁽⁵³⁸⁾ – pastes RVF from the Clipboard, if available;
- PasteRTF⁽⁵³⁷⁾ – pastes RTF from the Clipboard, if available;
- PasteHTML⁽⁵³⁶⁾ – pastes HTML from the Clipboard, if available;
- PasteURL⁽⁵³⁹⁾ – pastes URL from the Clipboard, if available [VCL, LCL, FMX for Windows];
- Paste⁽⁵³⁴⁾ – 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⁽⁴⁷⁰⁾ property is taken into account. This method is called automatically when user press **Ctrl + V** or **Shift + Insert**.

Testing is the Clipboard has the specific format:

- `CanPaste`⁽⁴⁹⁸⁾ – "does the Clipboard contain any format what can be pasted in RichViewEdit?"
- `CanPasteRVF`⁽⁴⁹⁹⁾ – "does the Clipboard contain RVF?"
- `CanPasteRTF`⁽⁴⁹⁹⁾ – "does the Clipboard contain RTF?"
- `CanPasteHTML`⁽⁴⁹⁹⁾ – "does the Clipboard contain HTML?"

`OnPaste`⁽⁵⁸⁴⁾ 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`⁽⁵⁰²⁾ copies the selection (like `CopyDef`) and then deletes it (like `DeleteSelection`⁽⁵⁰³⁾). 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⁽¹¹⁸⁾

3.2.5 Searching and replacing

The following main functions for text searching are available:

```
function TRichView(210).SearchTextW(346)(s: TRVUnicodeString(1032);
  SrchOptions: TRVSearchOptions(1024)): Boolean;
function TRichViewEdit(461).SearchTextW(543)(s: TRVUnicodeString(1032);
  SrchOptions: TRVESearchOptions(999)): Boolean;
```

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

- if `OnTextFound`⁽⁴¹⁰⁾ event is assigned, this event is called;
- otherwise, or if `DoDefault` parameter of `OnTextFound`⁽⁴¹⁰⁾ = `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`⁽⁴¹⁰⁾, 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`⁽³⁶⁷⁾ 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 GetRVSearchOptions987(fo: TFindOptions): TRVSearchOptions1024;
function GetRVESearchOptions986(fo: TFindOptions): TRVESearchOptions999;
```

It's easy to implement replacing of text in editor. SearchText selects the old substring, InsertText⁵³² 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**(CurItemNo⁴⁷²,...). But there is one important exception: when using tables¹⁹⁰, **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.ItemCount²³⁷ returns the number of items.

```
function RichView.GetItemStyle302(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.TextStyles⁶⁵⁴).

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
<i>rvsBreak</i>	-1	horizontal line ¹⁶⁷
<i>rvsPicture</i>	-3	picture ¹⁶³
<i>rvsHotspot</i>	-4	hotspot ¹⁷²
<i>rvsComponent</i>	-5	Delphi control ¹⁶⁸
<i>rvsBullet</i>	-6	bullet ¹⁷⁰
<i>rvsHotImage</i>	-10	hot-picture ¹⁶⁶
<i>rvsListMarker</i>	-11	list marker ¹⁷⁴ (bullet or numbering of paragraphs)
<i>rvsTab</i>	-12	tabulator ¹⁶²
<i>rvsTable</i>	-60	table ¹⁷³

See also: `rsvXXX` constants⁽¹⁰⁵⁹⁾)

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

- `GetBreakInfo`⁽²⁸⁹⁾, `GetBulletInfo`⁽²⁸⁹⁾, `GetHotspotInfo`⁽²⁹⁵⁾, `GetPictureInfo`⁽³¹⁰⁾, `GetControllInfo`⁽²⁹³⁾, `GetTextInfo`⁽³¹³⁾.
- For item of any type you can get additional property: `GetItemExtraIntProperty`⁽³⁰⁰⁾ and `GetItemExtraStrProperty`⁽³⁰⁰⁾.
- For item of any type you can get "tag" (`GetItemTag`⁽³⁰²⁾), "checkpoint" (`GetItemCheckpoint`⁽²⁹⁸⁾)
- For any item you can get information about its paragraph: (`GetItemPara`⁽³⁰¹⁾, `IsParaStart`⁽³¹⁸⁾, `IsFromNewLine`⁽³¹⁷⁾).
- For any item you can get its text (text for text item, name for non-text item) (`GetItemText`⁽³⁰³⁾)
- For any item you can get object representing it in a document (`GetItem`⁽²⁹⁶⁾, usually used for tables⁽⁸⁴⁶⁾)

In editor you can use the methods above or `GetCurrentBreakInfo`⁽⁵⁰⁴⁾, `GetCurrentBulletInfo`⁽⁵⁰⁵⁾, `GetCurrentHotspotInfo`⁽⁵⁰⁷⁾, `GetCurrentPictureInfo`⁽⁵¹³⁾, `GetCurrentControllInfo`⁽⁵⁰⁶⁾, `GetCurrentTextInfo`⁽⁵¹⁴⁾ (**GetCurrent***Info** methods), and `GetCurrentTag`⁽⁵¹⁴⁾, `GetCurrentCheckpoint`⁽⁵⁰⁶⁾, `GetCurrentItemText`⁽⁵¹¹⁾.

The arguments of these methods are similar to the arguments of **Add***Tag** methods for appending items⁽¹⁰²⁾.

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)⁽¹¹²⁾. Exception: items with `rvepShared`⁽¹⁰⁰⁰⁾ property.

A table at the position of the caret (or a top-level table containing the caret) can be obtained with the method `GetCurrentItemEx`⁽⁵⁰⁹⁾.

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`⁽⁴⁷²⁾,...) But there is one important exception – when using tables⁽¹⁹⁰⁾, **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⁽⁵⁰¹⁾ and vice versa⁽⁵⁰²⁾)

First, you need to know a type of item. You can use `RichView.GetItemStyle`⁽³⁰²⁾(`ItemNo`). Or you can use `CurItemStyle`⁽⁴⁷³⁾ 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 ⁽¹⁰²⁾.

For the specified items you can use

- `·SetBreakInfo` ⁽³⁵¹⁾, `SetBulletInfo` ⁽³⁵²⁾, `SetHotspotInfo` ⁽³⁵⁷⁾ (call `Invalidate` to repaint `RichView` after them, reformatting is not necessary)
- `·SetPictureInfo` ⁽³⁶³⁾, `SetControlInfo` ⁽³⁵⁴⁾ return Boolean value: "reformatting ⁽²⁸⁸⁾ 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` ⁽³⁶⁰⁾ method to change tags of items,
- `SetCheckpointInfo` ⁽³⁵³⁾ to change checkpoints of items,
- `SetItemExtraIntProperty` ⁽³⁵⁸⁾ to change values of additional integer properties,
- `SetItemExtraStrProperty` ⁽³⁵⁹⁾ to change values of additional string properties,
- `SetItemText` ⁽³⁶⁰⁾ to change item text (visible text for text items, names for non-text items).

Please read the topic about "tags" ⁽⁹¹⁾ if you want to change tags of items.

The limitations of this version are:

- you can't change `ImageList` of *bullets* ⁽¹⁷⁰⁾ and *hotspots* ⁽¹⁷²⁾;
- you can't replace control with the different one;

It is not recommended to use these methods in editor, see `Viewer vs Editor` ⁽¹³⁵⁾

Set***InfoEd methods (editor)

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

`SetBreakInfoEd` ⁽⁵⁴⁷⁾, `SetBulletInfoEd` ⁽⁵⁴⁸⁾, `SetHotspotInfoEd` ⁽⁵⁶⁰⁾, `SetPictureInfoEd` ⁽⁵⁶⁵⁾,
`SetControlInfoEd` ⁽⁵⁵⁰⁾;

also `SetItemTagEd` ⁽⁵⁶³⁾, `SetCheckpointInfoEd` ⁽⁵⁴⁹⁾, `SetItemTextEd` ⁽⁵⁶⁴⁾, `SetItemExtraIntPropertyEd` ⁽⁵⁶²⁾,
`SetItemExtraStrPropertyEd` ⁽⁵⁶²⁾

SetCurrent***Info methods (editor)

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

`SetCurrentBreakInfo` ⁽⁵⁵¹⁾, `SetCurrentBulletInfo` ⁽⁵⁵²⁾, `SetCurrentHotspotInfo` ⁽⁵⁵⁵⁾, `SetCurrentPictureInfo` ⁽⁵⁵⁹⁾,
`SetCurrentControlInfo` ⁽⁵⁵⁴⁾,

also `SetCurrentTag` ⁽⁵⁶⁰⁾, `SetCurrentCheckpointInfo` ⁽⁵⁵³⁾, `SetCurrentItemText` ⁽⁵⁵⁷⁾,
`SetCurrentItemExtraIntProperty` ⁽⁵⁵⁶⁾, `SetCurrentItemExtraStrProperty` ⁽⁵⁵⁷⁾.

Additional methods for controls

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

```
procedure ResizeControl (542)(ItemNo: Integer; NewWidth, NewHeight: TRVCoord (998));
procedure ResizeCurrentControl (543)(NewWidth, NewHeight: TRVCoord (998));
```

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 AdjustControlPlacement489(ItemNo: Integer);
procedure AdjustControlPlacement2489(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

Overview⁸⁴ | Editing

- TRichViewEdit: Inserting items at position of caret¹¹⁴
- Undo/redo in RichViewEdit¹¹⁵
- Drag&drop¹¹⁸

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 ReadOnly⁴⁸⁰ 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 OnChange⁵⁷² event, set Modified⁴⁷⁹ 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 CurTextStyleNo⁴⁷⁴ property):

```
procedure InsertText532(text, CaretBefore);
```

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

```
function InsertTextFromFile533(FileName, CodePage): Boolean;
```

```
function InsertTextFromFileW533(FileName): Boolean;
```

```
function InsertOEMTextFromFile525(FileName): Boolean;
```

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

```
function InsertStringTag530(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*** methods¹⁰²):

```
function InsertControl(517)(Name, ctrl, VAlign): Boolean;  
function InsertPicture(526)(Name, gr, VAlign): Boolean;  
function InsertHotPicture(519)(Name, gr, VAlign): Boolean;  
function InsertBreak(515)(Width, Style, Color): Boolean;  
function InsertBullet(515)(ImageIndex, ImageList): Boolean;  
function InsertHotspot(520)(ImageIndex, HotImageIndex, ImageList): Boolean;
```

Inserting Files and Streams

These methods insert RVF⁽¹²⁴⁾ file or stream

```
function InsertRVFFromStreamEd(529)(Stream): Boolean;  
function InsertRVFFromFileEd(528)(FileName): Boolean;
```

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

```
function InsertRTFFromStreamEd(528)(Stream): Boolean;  
function InsertRTFFromFileEd(527)(FileName): Boolean;
```

These methods insert text of the current text style (see CurTextStyleNo⁽⁴⁷⁴⁾ property):

```
function InsertTextFromFile(533)(FileName): Boolean;  
function InsertTextFromFileW(533)(FileName): Boolean;  
function InsertOEMTextFromFile(525)(FileName): Boolean;
```

Inserting Tables and Other Items

The most general method for inserting items is

```
function InsertItem(522)(Name, Item): Boolean;
```

It's recommended to use this method only for tables⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, numbered sequences⁽¹⁷⁷⁾, endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾, references to parent footnotes and endnotes⁽¹⁸⁴⁾, page numbers⁽¹⁸⁵⁾, page counts⁽¹⁸⁶⁾, equations⁽¹⁸⁷⁾, and custom item types⁽¹⁸⁹⁾.

See also...

Methods for pasting from the Clipboard⁽¹⁰⁸⁾.

3.3.2 Undo and redo

(Introduced in version 1.3)

All methods for document editing that introduced in editor (TRichViewEdit⁽⁴⁶¹⁾) can be undone/redone (see editor-style methods).

All methods for document modification that introduced in TRichView⁽²¹⁰⁾ 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(481): 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`⁽²⁷⁹⁾ method, or when you use methods for loading documents (which call `Clear` inside).

You can clear undo and redo buffer yourself using method `ClearUndo`⁽⁵⁰¹⁾.

Implementing menu/toolbar buttons for undo/redo

You can undo/redo operations from the application code, using methods `Undo`⁽⁵⁶⁷⁾ and `Redo`⁽⁵⁴⁰⁾.

You can change enabled/disabled state of menu items or buttons comparing the returned values of functions `UndoAction`⁽⁵⁶⁷⁾ and `RedoAction`⁽⁵⁴⁰⁾ with `rvutNone` (`rvutNone` – menu for undo/redo should be disabled; other values – enabled).

Call these methods inside `OnChange`⁽⁵⁷²⁾ 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`⁽⁵⁶⁷⁾/`RedoAction`⁽⁵⁴⁰⁾.

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`⁽⁵⁶⁸⁾/`RedoName`⁽⁵⁴⁰⁾ 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`⁽⁴⁹⁷⁾ to give a standard name, or `BeginUndoCustomGroup`⁽⁴⁹⁶⁾ to give a custom name.

Next, you need to call `SetUndoGroupMode`⁽⁵⁶⁷⁾.

```
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`⁽¹⁰³¹⁾, except for `rvutNone` and `rvutCustom`.

This value will be returned by `UndoAction/RedoAction` when undo/redo.

```
procedure TRichViewEdit.BeginUndoCustomGroup(const Name: TRVUnicodeString(1032));
```

– 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`⁽⁴⁸¹⁾)

▼ **Example (inserting picture and resizing it 100x100):**

```
MyRichViewEdit.TopLevelEditor.BeginUndoGroup(rvutInsert);
MyRichViewEdit.TopLevelEditor.SetUndoGroupMode(True);
try
  if MyRichViewEdit.InsertPicture(526)(...) then
    begin
      MyRichViewEdit.SetCurrentItemExtraIntProperty(556)(
        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`⁽⁴⁹⁷⁾ to give a standard name to the group, or use `BeginUndoCustomGroup2..EndUndoCustomGroup2`⁽⁴⁹⁶⁾ 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(365)(
  0, MyRichViewEdit.GetOffsBeforeItem(309)(0),
  0, MyRichViewEdit.GetOffsBeforeItem(0));
MyRichViewEdit.BeginUndoGroup2(rvutInsert);
try
  while MyRichViewEdit.SearchText(543)(FindText, [rvseoDown, rvseoMulti
    MyRichViewEdit.InsertText(532)(ReplaceText);
  finally
    MyRichViewEdit.EndUndoGroup2(rvutInsert);
  end;
```

See also...

Methods introduced in `TRichView` and undo: see `Viewer vs Editor`⁽¹³⁵⁾

3.3.3 Drag and drop

You can drag data from TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾. You can drop data to TRichViewEdit⁽⁴⁶¹⁾.

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⁽¹²⁴⁾ ('RichView Format')
- RTF ('Rich Text Format')
- ANSI text (CF_TEXT)
- Unicode⁽¹³⁰⁾ text (CF_UNICODETEXT)
- bitmap (CF_BITMAP), if TBitmap picture⁽¹⁶³⁾ is selected
- metafile (CF_ENHMETAFILE), if TMetafile picture⁽¹⁶³⁾ is selected.

You can disallow dragging from the given TRichView by including *rvoDisallowDrag* in the Options⁽²⁴⁰⁾ property.

TRichView cannot initialize dragging of inserted controls⁽¹⁶⁸⁾. If you want to drag them, use *BeginOleDrag*⁽²⁷⁸⁾ method.

Dragging to TRichViewEdit [VCL, LCL]

AcceptDragDropFormat⁽⁴⁷⁰⁾ lists formats that can be accepted.

By default all formats are accepted except for URL (because it requires processing in *OnReadHyperlink*⁽³⁸⁸⁾)

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⁽⁴⁷⁵⁾, 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⁽²⁴⁰⁾, file name is stored in *rvesplImageFileName* extra string property⁽¹⁰⁰⁵⁾ for pictures; you can modify this file name in *OnAssignImageFileName*⁽³⁷¹⁾ event.

The following events may occur:

- *OnReadHyperlink*⁽³⁸⁸⁾;
- *OnDropFile*⁽⁵⁷⁶⁾;
- *OnDropFiles*⁽⁵⁷⁶⁾.

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

- *OnOleDragEnter*⁽⁵⁷⁹⁾;
- *OnOleDragOver*⁽⁵⁸⁰⁾;
- *OnOleDragLeave*⁽⁵⁷⁹⁾;
- *OnOleDrop*⁽⁵⁸¹⁾,
- *OnBeforeOleDrop* and *OnAfterOleDrop*⁽⁵⁷⁰⁾

Drag&Drop in FireMonkey

When dragging between TRichView/TRichViewEdit controls within the same application, data are transferred using the most appropriate format from AcceptDragDropFormat⁽⁴⁷⁰⁾.

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⁽¹⁰⁸⁾

3.4 Graphics

Overview⁽⁸⁴⁾ | Graphics

- Animated images⁽¹¹⁹⁾
- Smooth image scaling⁽¹²⁰⁾
- Semitransparent objects⁽¹²¹⁾

3.4.1 Animated images

Overview

TRichView can animate inserted images⁽¹⁶³⁾.

AnimationMode⁽²²²⁾ property defines when animation starts. By default, AnimationMode⁽²²²⁾ = *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⁽²⁸³⁾. If you simply load and edit documents, set this mode to *rvaniOnFormat*.

Methods:

- StartAnimation⁽³⁶⁶⁾ starts all animations;
- StopAnimation⁽³⁶⁷⁾ stops all animations;
- ResetAnimation⁽³³³⁾ 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 (*rveplImageWidth* x *rveplImageHeight*) arranged in rows and columns. Animation is enabled if you set nonzero *rveplAnimationInterval* (animation delay in 1/100 of second), see TRVExtraltemProperty⁽¹⁰⁰⁰⁾ type. If at least one of *rveplImageWidth* and *rveplImageHeight* 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.BackgroundImage⁽²²²⁾ (if BackgroundStyle⁽²²³⁾=*bsStretched*);
- pictures⁽¹⁶³⁾ and *hot-pictures*⁽¹⁶⁶⁾ (when stretched);
- table.BackgroundImage⁽⁸⁴⁹⁾ (if BackgroundStyle⁽⁸⁵⁰⁾=*rvbsStretched*);
- table cells' BackgroundImage⁽⁸⁹⁶⁾ (if BackgroundStyle⁽⁸⁹⁷⁾=*rvbsStretched*).

The settings for scaling can be changed using RVThumbnailMaker⁽¹⁰⁶¹⁾ 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 *rveplSmoothScaling*⁽¹⁰⁰⁰⁾ property.

 **ScaleRichView note:** in TScaleRichViewEdit, 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⁽¹⁰⁴⁷⁾ scaled images (+ a scaled version TCustomRichView.BackgroundImage⁽²²²⁾, 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 ⁽¹⁶³⁾ (for example, Gif, Png, icons, Bitmaps with alpha-channel)
- paragraph background (with specified Opacity ⁽⁷⁴²⁾)
- table ⁽¹⁷³⁾ background (that may include transparent pictures ⁽⁸⁴⁹⁾ and semitransparent ⁽⁸⁶⁰⁾ color filling)
- table cell background (that may include transparent pictures ⁽⁸⁹⁶⁾ and semitransparent ⁽⁹⁰²⁾ color filling)
- sidenote ⁽¹⁸¹⁾ and text box ⁽¹⁸³⁾ background (with specified Opacity), displayed only while printing in TRVPrint ⁽⁷⁵⁹⁾ and in ScaleRichView.

FireMonkey note: in FireMonkey, any colors can be semitransparent, so any object can be semitransparent. Opacity properties are applied in addition 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 ⁽¹¹⁹⁾ and backgrounds of inplace editors of table cells.

Known problems: TPngObject does not support semitransparency when displayed rotated ⁽⁹⁰⁴⁾.

Printing in TRVPrint ⁽⁷⁵⁹⁾ [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 ⁽⁸⁰⁴⁾

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 ⁽⁸⁰⁶⁾ 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 ⁽⁸⁴⁾ | Saving and Loading

- `TRichView` methods for saving and loading ⁽¹²²⁾
- RichView Format (RVF) Overview ⁽¹²⁴⁾
- Export to HTML ⁽¹²⁷⁾

3.5.1 Saving and loading

TRichView has the following methods for export:

- `SaveHTML` ⁽³³⁵⁾, `SaveHTMLToStream` ⁽³³⁸⁾ – exporting the document to a file or a stream as HTML;
- `SaveText` ⁽³⁴⁵⁾: saving plain ANSI text in file; `SaveTextW` ⁽³⁴⁵⁾: saving plain Unicode text; `SaveTextToStream`, `SaveTextToStreamW` ⁽³⁴⁵⁾ – the same for streams (can save the whole document or only a selection);
- `SaveMarkdown` ⁽³⁴⁰⁾, `SaveMarkdownToStream` ⁽³⁴¹⁾ – saving the whole document or the selected part to **Markdown** file or stream;
- `SaveRTF` ⁽³⁴³⁾, `SaveRTFToStream` ⁽³⁴⁴⁾ – exporting the whole document or the selected part to a file or a stream as RTF (Rich Text Format);
- `SaveDocX` ⁽³³³⁾, `SaveDocXToStream` ⁽³³⁴⁾ – exporting the whole document or the selected part to a file or a stream as DocX (Microsoft Word format);
- `SaveRVF` ⁽³⁴⁴⁾, `SaveRVFToStream` ⁽³⁴⁴⁾ – saving the whole document or the selected part to RVF (RichView Format ⁽¹²⁴⁾) file or stream.

TRichView has the following methods for import:

- `LoadHTML` ⁽³²²⁾, `LoadHTMLFromStream` ⁽³²⁴⁾ – reading HTML file or stream;
- `LoadText` ⁽³³⁰⁾: reading plain ANSI text; `LoadTextW` ⁽³³⁰⁾: reading plain Unicode (UTF-16) text;
- `LoadTextFromStream`, `LoadTextFromStreamW` ⁽³³¹⁾ – the same for streams instead of files;
- `LoadMarkdown` ⁽³²⁵⁾, `LoadMarkdownFromStream` ⁽³²⁵⁾: reading **Markdown** file or stream;
- `LoadRTF` ⁽³²⁶⁾, `LoadRTFFromStream` ⁽³²⁷⁾ – reading RTF (Rich Text Format) file or stream.

- LoadDocX⁽³¹⁸⁾, LoadDocXFromStream⁽³¹⁹⁾ – reading DocX (Microsoft Word Document) file or stream.
- LoadRVF⁽³²⁸⁾, LoadRVFFromStream⁽³²⁹⁾ – reading RVF (RichView Format⁽¹²⁴⁾) file or stream.

Universal methods:

- LoadFromFile⁽³²⁰⁾, LoadFromStream⁽³²¹⁾ – reading from a file or a stream, a format is detected by content.

See also: InsertRVFFromStream⁽³¹⁶⁾, AppendRVFFromStream⁽²⁷⁶⁾.

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

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

- InsertRVFFromFileEd⁽⁵²⁸⁾, InsertRVFFromStreamEd⁽⁵²⁹⁾ – like InsertRVFFromFile/InsertRVFFromStream, but editor-style methods.
- InsertRTFFromFileEd⁽⁵²⁷⁾, InsertRTFFromStreamEd⁽⁵²⁸⁾ – inserting RTF (Rich Text Format) file or stream.
- InsertDocXFromFileEd⁽⁵¹⁸⁾, InsertDocXFromStreamEd⁽⁵¹⁸⁾ – inserting DocX (Microsoft Word Document) file or stream.
- InsertTextFromFile⁽⁵³³⁾ – editor-style method;
InsertOEMTextFromFile⁽⁵²⁵⁾ – the same method for files in DOS code page;
InsertTextFromFileW⁽⁵³³⁾ – the same method for Unicode files;
- InsertMarkdownFromFileEd⁽⁵²³⁾, InsertMarkdownFromStreamEd⁽⁵²⁴⁾ – inserting **Markdown** file or stream.

There are also method for pasting data from the Clipboard⁽¹⁰⁸⁾.

Events on export

- HTML: OnHTMLSaveImage⁽³⁷⁵⁾, OnSaveImage2⁽⁴⁰²⁾, OnSaveComponentToFile⁽³⁹⁷⁾, OnSaveHTMLExtra⁽⁴⁰¹⁾, OnSaveItemToFile⁽⁴⁰⁴⁾, OnSaveParaToHTML⁽⁴⁰⁶⁾, OnWriteHyperlink⁽⁴¹¹⁾, OnWriteObjectProperties⁽⁴¹³⁾.
- RTF: OnSaveComponentToFile⁽³⁹⁷⁾, OnSaveRTFExtra⁽⁴⁰⁶⁾, OnSaveItemToFile⁽⁴⁰⁴⁾, OnWriteHyperlink⁽⁴¹¹⁾, OnWriteObjectProperties⁽⁴¹³⁾.
- DocX: OnSaveComponentToFile⁽³⁹⁷⁾, OnSaveDocXExtra⁽³⁹⁹⁾, OnSaveItemToFile⁽⁴⁰⁴⁾, OnWriteHyperlink⁽⁴¹¹⁾, OnWriteObjectProperties⁽⁴¹³⁾.
- Text: OnSaveComponentToFile⁽³⁹⁷⁾, OnSaveItemToFile⁽⁴⁰⁴⁾.
- Markdown: OnSaveImage2⁽⁴⁰²⁾, OnSaveComponentToFile⁽³⁹⁷⁾, OnWriteHyperlink⁽⁴¹¹⁾.

Events on import

- Markdown: OnImportPicture⁽³⁷⁸⁾, OnReadHyperlink⁽³⁸⁸⁾, OnAddStyle⁽³⁷⁰⁾.
- RTF, DocXL: the events above + OnStyleTemplatesChange⁽⁴¹⁰⁾.
- HTML: the events above + OnImportFile⁽³⁷⁸⁾.

Common events

- OnProgress⁽³⁸⁵⁾

See also...

Saving HTML⁽¹²⁷⁾

Reading/writing RVF⁽¹²⁴⁾

Export of tables⁽²⁰³⁾

Unicode in TRichView⁽¹³⁰⁾

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⁽²⁴⁷⁾).

This format is used:

- in TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ to save their contents to binary fields;
- RVF can be copied to the Clipboard; editor can paste RVF;
- data can be dragged&dropped⁽¹¹⁸⁾ as RVF;
- you can load/save RVF with the methods: LoadRVFFromStream⁽³²⁹⁾, InsertRVFFromStream⁽³¹⁶⁾, AppendRVFFromStream⁽²⁷⁶⁾, LoadRVF⁽³²⁸⁾, SaveRVFToStream⁽³⁴⁴⁾, SaveRVF⁽³⁴⁴⁾.

All TRichView 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⁽²¹⁸⁾;
- in order to load *bullets*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾ and list levels⁽⁷⁵⁰⁾ with imagelist from RVF you need to process OnRVFImageListNeeded⁽³⁹³⁾ event;
- you can save pictures⁽¹⁶³⁾ and controls⁽¹⁶⁸⁾ 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⁽⁶⁵⁴⁾ and paragraph⁽⁶⁴⁸⁾ (i.e. styles) is saved in RVF documents optionally, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions⁽²⁴⁷⁾;
- style templates are saved in RVF, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions⁽²⁴⁷⁾, and UseStyleTemplates⁽²⁵⁶⁾ = True.

RVF and Image Lists

TRichView can save and load all document content itself, but there is an exception: *bullets*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾ and list markers having imagelist picture in list levels⁽⁷⁵⁰⁾.

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⁽³⁹³⁾ 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*⁽¹⁷⁰⁾ or *hotspot*⁽¹⁷²⁾. 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⁽⁹¹⁾. 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⁽³⁹³⁾ and OnRVFControlNeeded⁽³⁹³⁾ events.

▼ Example

```
procedure TMyForm.MyRichViewRVFControlNeeded(  
  Sender: TCustomRichView; Name: TRVUnicodeString(1032);  
  const Tag: TRVTag(1029); 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⁽¹⁰²⁹⁾ – TRichView item tag (do not confuse with tag properties of TComponent descendants);
- Name: TRVUnicodeString⁽¹⁰³²⁾ – 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(210);
  const ItemName: TRVUnicodeString(1032); Item: TRVNonTextItemInfo(844);
  Index1, Index2: Integer; var gr: TRVGraphic(970));
begin
  gr := TBitmap.Create;
  if (Index1>=0) and (Index2>=0) then
    gr.LoadFromFile((Item as TRVTableItemInfo(846)).Cells(857)[Index1, Index
  else if Item is TRVTableItemInfo then
    gr.LoadFromFile(TRVTableItemInfo(Item).BackgroundImageFileName(850))
  else if Item is TRVGraphicItemInfo then
    gr.LoadFromFile(TRVGraphicItemInfo(844)(Item).ImageFileName(1005));
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 *rvfolgnoreGraphicClasses* in RichView.RVFOptions⁽²⁴⁷⁾, 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⁽²⁴⁷⁾ define options for RVF saving and loading.

After loading you can check RichView.RVFWarnings⁽²⁵¹⁾ 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⁽¹³⁹⁾) than specified in Style⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾ of TRichView which loads this file.

When inserting RVF, lengths from RVF are converted to Style⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾.

When loading RVF, the control works according to *rvfoCanChangeUnits* option. If it is excluded from RVFOptions⁽²⁴⁷⁾, lengths from RVF are converted to Style⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾. If it is included, and RVF contains either text or paragraph styles, existing styles are converted to units read from RVF (using Style⁽⁶³⁰⁾.ConvertToDifferentUnits⁽⁶⁵⁷⁾ method), so lengths from RVF are read without conversion.

If a conversion of units is performed while reading RVF, *rvfwConvUnits* is included in *RVFWarnings* ⁽²⁵¹⁾.

See also...

RVF Specification ⁽¹⁴³⁾

RichView properties:

- Options ⁽²⁴⁰⁾,
- RVFOptions ⁽²⁴⁷⁾,
- RVFWarnings ⁽²⁵¹⁾;
- UseStyleTemplates ⁽²⁵⁶⁾;
- StyleTemplateInsertMode ⁽²⁵⁴⁾.

RichView events:

- OnRVFImageListNeeded ⁽³⁹³⁾,
- OnVFControlNeeded ⁽³⁹²⁾,
- OnRVFPictureNeeded ⁽³⁹³⁾,
- OnControlAction ⁽³⁷³⁾ (ControlAction=*rvcaAfterRVFLoad*),
- OnItemAction ⁽³⁸⁰⁾ (ItemAction=*rvialInserting* and *rvialInserted*);
- OnStyleTemplatesChange ⁽⁴¹⁰⁾.

RichView methods:

- LoadRVFFromStream ⁽³²⁹⁾, InsertRVFFromStream ⁽³¹⁶⁾, AppendRVFFromStream ⁽²⁷⁶⁾, LoadRVF ⁽³²⁸⁾,
- SaveRVFToStream ⁽³⁴⁴⁾, SaveRVF ⁽³⁴⁴⁾,
- Copy ⁽²⁸¹⁾, CopyDef ⁽²⁸¹⁾, CopyRVF ⁽²⁸²⁾.

RichViewEdit methods:

- InsertRVFFromStreamEd ⁽⁵²⁹⁾, InsertRVFFromFileEd ⁽⁵²⁸⁾,
- Paste ⁽⁵³⁴⁾, PasteRVF ⁽⁵³⁸⁾, CanPasteRVF ⁽⁴⁹⁹⁾, CutDef ⁽⁵⁰²⁾.

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 ⁽²³⁷⁾
- DocParameters ⁽²³⁰⁾.Title ⁽⁴¹⁹⁾ and Comments ⁽⁴¹⁷⁾.

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 ⁽¹⁰⁵⁷⁾.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`⁽³⁷⁵⁾ or `OnSaveImage2`⁽⁴⁰²⁾ event (see demo in `Demos*\Assorted\Save HTML`).

See also `HTMLSaveProperties`⁽²³⁷⁾.`ImageOptions`⁽⁴³⁴⁾, `ImagesPrefix`⁽⁴³⁶⁾.

Two ways of HTML saving

There are two ways to save HTML in TRichView, defined `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾:

- saving using ``, ``, `<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`⁽⁸⁷⁾ are saved as anchors.

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

See `HTMLSaveProperties`⁽²³⁷⁾.`UseCheckpointNames`⁽⁴³⁷⁾, `CheckpointsPrefix`⁽⁴³⁰⁾.

Encoding

By default, TRichView creates UTF-8 HTML files.

UTF-8 is highly recommended, but you can change encoding using `HTMLSaveProperties`⁽²³⁷⁾.`Encoding`⁽⁴³¹⁾ property.

Saving Using CSS

By default, if `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *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`⁽²³⁷⁾.`ExternalCSSFileName`⁽⁴³²⁾.

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`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *rvhtmlstSimplified*), you can store non-text items (such as tables, images or horizontal lines) with CSS.

See `HTMLSaveProperties`⁽²³⁷⁾.`CSSOptions`⁽⁴³⁰⁾.

You can provide additional styles in `HTMLSaveProperties`⁽²³⁷⁾.`ExtraStyles`⁽⁴³³⁾.

Hyperlinks, controls, more...

By default, items tags⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using the event `OnWriteHyperlink`⁽⁴¹¹⁾.

By default, inserted controls are not saved. However, you can save them in the event `OnSaveComponentToFile`⁽³⁹⁷⁾.

You can save additional information in HTML in the events: `OnSaveHTMLExtra`⁽⁴⁰¹⁾, `OnSaveParaToHTML`⁽⁴⁰⁶⁾.

You can insert HTML codes⁽⁷⁰⁴⁾ in text.

You can completely change saving of items of any type in HTML using `OnSaveItemToFile`⁽⁴⁰⁴⁾ event.

You can save additional properties of non-text items using `OnWriteObjectProperties`⁽⁴¹³⁾ event.

Equation objects

If the free version of Adit Math Engine is used, equation objects⁽¹⁸⁷⁾ are exported as images.

If the free commercial of Adit Math Engine is used, equation objects⁽¹⁸⁷⁾ are exported as MathML objects (can be turned off by `RVMathEquationManager`⁽⁹⁹¹⁾.`SaveMathMLInHTML`⁽⁹⁶⁹⁾ property).

Symbol Fonts

`SaveHTML`⁽³³⁵⁾ 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⁽³⁵⁶⁾ and/or a footer⁽³⁵⁵⁾ 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`⁽²³⁰⁾.`TitlePage`⁽⁴¹⁹⁾.

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⁽¹²²⁾;
- Export of tables⁽²⁰³⁾;
- Typed constants related to HTML export⁽¹⁰⁴⁰⁾;
- 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⁽⁸⁴⁾ | Internationalization

- Unicode⁽¹³⁰⁾
- Bi-directional text⁽¹³²⁾

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⁽³³⁰⁾, LoadTextFromStream⁽³³¹⁾ load ANSI text files. A code page for conversion to Unicode is specified in the optional parameter.

LoadTextW⁽³³⁰⁾, LoadTextFromStreamW⁽³³¹⁾ load Unicode text files.

Note: you can test file with the function

```
function RV_TestFileUnicode(const FileName: TRVUnicodeString(1032)): TRVUnicodeTestRe  
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⁽³⁴⁵⁾ saves ANSI text file. Unicode strings are converted basing on Style.DefCodePage⁽⁶³⁵⁾ property.

SaveTextW⁽³⁴⁵⁾ 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***⁽¹²⁷⁾ 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⁽³¹³⁾ returns selection as an ANSI string. Unicode text is converted basing on Style.DefCodePage⁽⁶³⁵⁾ property.

GetSelTextW⁽³¹³⁾ returns selection as a Unicode string.

Text searching methods have versions allowing to search for ANSI and for Unicode string: TRichView.SearchTextA/SearchTextW⁽³⁴⁶⁾; however, SearchTextA simply converts the string to Unicode (using Style.DefCodePage⁽⁶³⁵⁾) and calls SearchTextW.

CopyTextA⁽²⁸³⁾ copies selection as ANSI text. Unicode strings are converted basing on Style.DefCodePage⁽⁶³⁵⁾ property.

CopyTextW⁽²⁸³⁾ copies selection as Unicode.

Copy⁽²⁸¹⁾ and CopyDef⁽²⁸¹⁾ are copy Unicode (Option⁽²⁴⁰⁾-rvoAutoCopyUnicodeText)

Editing Operations

If pasting text using Paste⁽⁵³⁴⁾ method, and text is available in Clipboard, the method pastes Unicode text.

PasteTextA⁽⁵³⁸⁾ pastes ANSI text, PasteTextW⁽⁵³⁸⁾ pastes Unicode text.

InsertTextFromFile⁽⁵³³⁾: the file must be ANSI (converted, if needed)

InsertOEMTextFromFile⁽⁵²⁵⁾: the file must be OEM (converted, if needed)

InsertTextFromFileW⁽⁵³³⁾: the file must be Unicode (converted, if needed)

InsertText⁽⁵³²⁾, InsertStringTag⁽⁵³⁰⁾ add Unicode string in Delphi/C++Builder 2009+ and ANSI string in older versions of Delphi/C++Builder.

InsertTextA⁽⁵³²⁾, InsertStringATag⁽⁵³⁰⁾ add ANSI string (converted, if needed)

InsertTextW⁽⁵³²⁾, InsertStringWTag⁽⁵³⁰⁾ add Unicode string (converted, if needed)

RVF (RichView Format⁽¹²⁴⁾)

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⁽²⁵¹⁾: *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 ⁽⁴⁶⁰⁾.

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` ⁽²²⁴⁾ property of `TRichView` either to `rvbdLeftToRight` or to `rvbdRightToLeft`.

`BiDiMode` of document can be overridden by `BiDiMode` of paragraph ⁽⁷²¹⁾, `BiDiMode` of paragraph can be overridden by `BiDiMode` of text item ⁽⁶⁹⁹⁾.

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` ⁽⁶⁵⁴⁾:

- 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 in 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 ⁽⁸⁴⁾ | Spelling Check

- Live spelling check ⁽¹³²⁾

3.7.1 Live spelling check

A live spelling check marks misspelled words with special underlines:

- [VCL and LCL] wavy line of `TRVStyle.LiveSpellingColor` ⁽⁶⁴⁶⁾ color
- [FMX] dashed line of `TRVStyle.LiveSpellingColor` ⁽⁶⁴⁶⁾ color

A spelling check is executed in background thread.

Starting and finishing spelling check

In `TRichView` ⁽²¹⁰⁾:

Spelling check is started by calling `StartLiveSpelling` ⁽³⁶⁷⁾ method.

In `TRichViewEdit` ⁽⁴⁶¹⁾:

Spelling check is started:

- by calling `StartLiveSpelling` ⁽³⁶⁷⁾ method or
- when user modifies document or when any editing-style method is called (if `LiveSpellingMode` ⁽⁴⁷⁹⁾ = `rvlspOnChange`)

Spelling check is stopped by `ClearLiveSpellingResults` ⁽²⁷⁹⁾ method. `Clear` ⁽²⁷⁹⁾ also stops checking.

Using spelling checking components



The recommended component for spelling checking is `TRVSpellChecker` ⁽⁷⁸⁴⁾.

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` ⁽⁴⁰⁹⁾ event for each word in the document.

“Word” is defined as a run of characters between delimiters ⁽²²⁸⁾. Additionally, ‘&’ is treated as a delimiter. Apostrophes are processed specially, see `RichViewApostropheInWord` ⁽¹⁰⁴³⁾ global variable.

URLs are ignored, see `RVIsCustomURL` ⁽¹⁰⁵⁸⁾ 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` ⁽⁴⁷²⁾ = `True`.

In this case, it's not necessary to process `OnSpellingCheck` ⁽⁴⁰⁹⁾ 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` ⁽³¹⁸⁾ if user added word in a custom dictionary or ignore list.

Call `GetCurrentMisspelling` ⁽⁵¹³⁾ 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` ⁽⁵¹³⁾ to get misspelled word in the caret position and to generate suggestions;

- when the user has chosen a suggestion, use `GetCurrentMisspelling`⁽⁵¹³⁾ again to select the misspelled word and use `InsertText`⁽⁵³²⁾ 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`⁽³⁷⁵⁾ 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`⁽²⁷⁹⁾ and then create a new document⁽¹⁰²⁾, everything is OK, because Clear stops live spelling check. But if you make changes in existing document, call `ClearLiveSpellingResults`⁽²⁷⁹⁾ 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⁽⁸⁴⁾ | Technical Information

- Viewer vs editor⁽¹³⁵⁾
- Controls, documents, items⁽¹³⁶⁾
- Valid documents⁽¹³⁸⁾
- Units of measurement⁽¹³⁹⁾
- Custom drawing⁽¹⁴¹⁾
- Cursors⁽¹⁴²⁾
- RichView Format (RVF) specification⁽¹⁴³⁾
- Lazarus⁽¹⁵⁴⁾
- FireMonkey⁽¹⁵⁵⁾
- Skia and Skia4Delphi⁽¹⁵⁷⁾

3.8.1 Viewer vs editor

Methods of TRichView

TRichView⁽²¹⁰⁾ 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⁽²⁷⁰⁾, AddPicture⁽²⁷²⁾, LoadRTF⁽³²⁶⁾.

Examples of the second group are SetPictureInfo⁽³⁶³⁾, SetItemText⁽³⁶⁰⁾.

Examples of the third group are: LoadRVF⁽³²⁸⁾, LoadRTFFromStream⁽³²⁷⁾.

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⁽²⁸⁸⁾ (or FormatTail⁽²⁸⁸⁾) to display documents created or modified with such methods.

Table cells⁽⁸⁹⁵⁾ 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⁽²⁷⁹⁾ or ClearLiveSpellingResults⁽²⁷⁹⁾).

Methods of TRichViewEdit

TRichViewEdit⁽⁴⁶¹⁾ 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⁽⁵³²⁾, InsertPicture⁽⁵²⁶⁾);
3. methods for modifying the item at the position of caret (**SetCurrent***** methods, such as SetCurrentPictureInfo⁽⁵⁵⁹⁾, SetCurrentItemText⁽⁵⁵⁷⁾);
4. extended set of methods for modifying item with the given index (**Set***Ed** methods, such as SetPictureInfoEd⁽⁵⁶⁵⁾, SetItemTextEd⁽⁵⁶⁴⁾).

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⁽⁴⁸⁰⁾ they do nothing; they also respect protection of text⁽⁷⁰⁵⁾ and paragraphs⁽⁷²³⁾;
- they call OnChange⁽⁵⁷²⁾ event;

- they can be safely called while a live spelling is running;
- modifications can be undone and redone⁽¹¹⁵⁾.

In an editor, the main application of the methods of the group 1 is generating new documents from scratch (after calling Clear⁽²⁷⁹⁾).

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⁽²⁷⁹⁾; (this method also clears undo and redo buffers);
2. One or several calls of methods of the group 1;
3. Format⁽²⁸⁸⁾ (after the document has been formatted, you can apply methods of the groups 2-4)

See also: Undo in tables⁽²⁰²⁾

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⁽³³⁾.

The main visual controls are: TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾.

2. Documents, objects containing items, descendants of TCustomRVFormattedData⁽⁹⁵⁷⁾.

Documents are objects, but not components.

Each RichView or RichViewEdit has such document accessible as RVData⁽²⁴⁷⁾ 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⁽⁸⁴⁶⁾ have cells⁽⁸⁹⁵⁾. Cells are also documents containing items, cells are descendants of TCustomRVFormattedData.

3. Items of documents, descendants of TCustomRVItemInfo⁽⁸⁴⁴⁾.

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⁽⁸⁴⁶⁾).

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⁽⁹⁰⁷⁾ (if cell is not edited, it is the cell itself; if edited, it is RVData⁽²⁴⁷⁾ 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(957));
var i,r,c: Integer;
    s: String;
    table: TRVTableItemInfo(846);
begin
for i := 0 to RVData.ItemCount(237)-1 do
  if RVData.GetItemStyle(302)(i)>=0 then
    begin
      // this is a text item
      s := RVData.GetItemText(303)(i);
      s := AnsiUpperCase(s);
      RVData.SetItemText(360)(i,s);
    end
  else if RVData.GetItemStyle(302)(i) is TRVTableItemInfo(846) then
    begin
      table := TRVTableItemInfo(846)(RVData.GetItem(296)(i));
      for r := 0 to table.RowCount(864)-1 do
        for c := 0 to table.ColCount(858)-1 do
          if table.Cells(857)[r,c]<>nil then
            AllUpperCase(table.Cells(857)[r,c].GetRVData(907));
          end;
    end;
end;

```

Call:

```

AllUpperCase(MyRichView.RVData);
MyRichView.Format(288);

```

Example 2

The same task, using different methods

```

// This procedure will be called for each RichView item
procedure TForm1.EnumItemsProc(RVData: TCustomRVData(957);
  ItemNo: Integer; var UserData1: Integer;
  const UserData2: TRVUnicodeString(1032); var ContinueEnum: Boolean);
var s: String;
begin
  if RVData.GetItemStyle(302)(ItemNo)>=0 then begin
    s := RVData.GetItemText(303)(i);
    s := AnsiUpperCase(s);
    RVData.SetItemText(360)(ItemNo,s);
  end;

```

```

end;
ContinueEnum := True;
end;
Call:
var v: Integer;

v := 0;
// RVDData.EnumItems calls the specified procedure (EnumItems)
// for each items in RVDData 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.RVDData(247).EnumItems(EnumItemsProc, v, '');
MyRichView.Format(288);

```

The advantage of using RVDData.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⁽¹⁵⁸⁾ 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⁽⁴⁶¹⁾ always create a valid document (providing that it was valid before the operation).

But some operations in viewer⁽²¹⁰⁾ can create invalid documents (especially DeletelItems⁽²⁸³⁾)

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 DeletelItems⁽²⁸³⁾ and SetAddParagraphMode⁽³⁵⁰⁾)
2. The first item (in document or table cell) must start a paragraph.
3. Full-line items (*breaks*⁽¹⁶⁷⁾ and *tables*⁽¹⁷³⁾) 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⁽¹⁷⁴⁾ 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⁽⁷⁰¹⁾ >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 ⁽⁷⁰⁷⁾.

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 ⁽⁶⁵⁵⁾ property of TRVStyle ⁽⁶³⁰⁾ component. When rendering on a device (screen or paper), sizes are recalculated according to the device DPI as:

$$\text{value} * (\text{device DPI}) / \text{UnitsPixelsPerInch} \supset (655)$$

Screen DPI can be read from Screen.PixelsPerInch, or it can be overridden in RichViewPixelsPerInch ⁽¹⁰⁴⁷⁾ global variable.

Properties measured in points

Font size (Size ⁽⁷⁰⁷⁾ property of text styles ⁽⁶⁵⁴⁾)

Properties measured in “standard” pixels

1 “standard” pixel = 1/TRVStyle.UnitsPixelsPerInch ⁽⁶⁵⁵⁾ of an inch.

The following properties of TCustomRichView ⁽²¹⁰⁾ are measured in “standard” pixels:

- LeftMargin ⁽²³⁸⁾, RightMargin ⁽²⁴⁴⁾, TopMargin ⁽²⁵⁵⁾, BottomMargin ⁽²²⁵⁾ – margins;
- MaxTextWidth ⁽²³⁹⁾ – maximal width for word wrapping;
- MinTextWidth ⁽²⁴⁰⁾ – minimal width for word wrapping (minimal width of scrollable area, not including margins).

Properties measured in TRVUnits ⁽¹⁰³²⁾

Properties of the type TRVLength ⁽¹⁰¹⁵⁾ are measured in TRVUnits ⁽¹⁰³²⁾. 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⁽²¹⁰⁾.DocParameters⁽²³⁰⁾; they are measured in TRichView⁽²¹⁰⁾.DocParameters⁽²³⁰⁾.Units⁽⁴²⁰⁾.
- properties of TRVPrint⁽⁷⁵⁹⁾; they are measured in TRVPrint⁽⁷⁵⁹⁾.Units⁽⁷⁷⁰⁾.

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⁽¹⁰²⁷⁾

Properties of the type TRVStyleLength⁽¹⁰²⁷⁾ are measured in TRVStyleUnits⁽¹⁰²⁷⁾. They are values of an integer type that can represent lengths measured in “standard” pixels, twips, or EMU.

The property TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾ defines units for properties (and sub-properties) of this TRVStyle component as well as for properties of items⁽¹⁵⁸⁾ in documents loaded in linked⁽²⁵³⁾ TRichView controls.

Properties of TRVStyle⁽⁶³⁰⁾ measured in TRVStyleUnits⁽¹⁰²⁷⁾

The following properties are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- DefTabWidth⁽⁶³⁶⁾;
- CharSpacing⁽⁷⁰⁰⁾ of text style⁽⁷¹²⁾;
- FirstIndent⁽⁷²²⁾, LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾, SpaceBefore⁽⁷²⁶⁾, SpaceAfter⁽⁷²⁶⁾ of paragraph style⁽⁷²⁷⁾;
- Position⁽⁷⁴⁸⁾ of item in a tab stop collection⁽⁷²⁶⁾ of paragraph style;
- Left, Right, Top, Bottom of background padding rectangle (Background⁽⁷²¹⁾.BorderOffsets⁽⁷⁴²⁾) of paragraph style;
- Left, Right, Top, Bottom of border padding rectangle (Border⁽⁷²¹⁾.BorderOffsets⁽⁷⁴³⁾) of paragraph style, as well as border's Width⁽⁷⁴⁴⁾ and InternalWidth⁽⁷⁴⁴⁾;
- MarkerIndent⁽⁷⁵⁷⁾, FirstIndent⁽⁷⁵¹⁾, LeftIndent⁽⁷⁵⁵⁾ in list levels⁽⁷³²⁾ of list style⁽⁷³¹⁾.

Item⁽¹⁵⁸⁾ properties measured in TRVStyleUnits⁽¹⁰²⁷⁾

The following items properties⁽¹⁰⁰⁰⁾ are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- *rvepSpacing*, *rveplmageWidth*, *rveplmageHeight*, *rvepVShift* (the latter only if *rvepVShiftAbs* is nonzero)

The following table⁽¹⁷³⁾ properties are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- BorderWidth⁽⁸⁵³⁾, CellBorderWidth⁽⁸⁵⁵⁾;
- BorderHSpacing⁽⁸⁵¹⁾, BorderVSpacing⁽⁸⁵³⁾, CellHSpacing⁽⁸⁵⁶⁾, CellVSpacing⁽⁸⁵⁷⁾;
- CellHPadding⁽⁸⁵⁵⁾, CellVPadding⁽⁸⁵⁷⁾, CellPadding⁽⁸⁵⁶⁾;
- HRuleWidth⁽⁸⁶⁰⁾, VRuleWidth⁽⁸⁶⁷⁾;
- BestWidth⁽⁸⁵⁰⁾ (only if value is positive).

The following table cell⁽⁸⁹⁵⁾ properties are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- BestHeight⁽⁸⁹⁸⁾;
- BestWidth⁽⁸⁹⁸⁾ (only if value is positive).

A width of break⁽¹⁶⁷⁾ (horizontal line) is measured in TRVStyleUnits⁽¹⁰²⁷⁾ as well.

The following label item⁽¹⁷⁶⁾ properties are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- MinWidth⁽⁹¹⁴⁾.

The following sidenote⁽¹⁸¹⁾ properties are measured in TRVStyleUnits⁽¹⁰²⁷⁾:

- BoxProperties⁽⁹³⁵⁾.Width⁽⁹³⁹⁾ and Height⁽⁹³⁸⁾ (if WidthType and HeightType are “absolute”);
- BoxPosition⁽⁹³⁵⁾.HorizontalOffset⁽⁹⁴²⁾ and VerticalOffset⁽⁹⁴⁶⁾ (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⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾.

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⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾=*rvstuPixels*), or as fractional values in points (if TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾=*rvstuTwips*).

In RVF⁽¹²⁴⁾ (RichView Format) files, values are saved as TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾. When loading, they are either converted to TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾, or vice versa, TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾ is changed. See “Units of measurement” in RVF Overview⁽¹²⁴⁾. Alternatively, TRichView can be converted to units from RVF file, if *rvfoCanChangeUnits* is included in RVFOptions⁽²⁴⁷⁾.

3.8.5 Custom drawing

There are the following events for custom drawing.

TCustomRichView⁽²¹⁰⁾

- OnPaint⁽³⁸⁴⁾ – drawing over RichView document.

TRVPrint⁽⁷⁵⁹⁾

- OnPagePrepaint⁽⁷⁸²⁾ – drawing on page under the document;
- OnPagePostpaint⁽⁷⁸¹⁾ – drawing on page over the document;
- OnPrintComponent⁽⁸³¹⁾ – printing components.

TRVStyle⁽⁶³⁰⁾

- OnDrawCheckpoint⁽⁶⁶⁷⁾ – drawing checkpoint;
- OnDrawPageBreak⁽⁶⁶⁸⁾ – drawing page break;
- OnDrawParaBack⁽⁶⁷⁰⁾ – drawing background of paragraph;
- OnDrawTextBack⁽⁶⁷³⁾ – drawing background of text;
- OnApplyStyle⁽⁶⁶⁵⁾ – applying text style to Canvas;
- OnApplyStyleColor⁽⁶⁶⁶⁾ – applying the text style color to Canvas;
- OnDrawStyleText⁽⁶⁷¹⁾ – drawing text of the given style;
- OnStyleHoverSensitive⁽⁶⁷⁴⁾ – returns "should component be repainted when user moves the mouse to a hypertext of the given style?"

TRVTableItemInfo⁽⁸⁴⁶⁾

- OnDrawBorder⁽⁸⁹³⁾ – drawing table and cell borders.

3.8.6 Cursors

Definition and use

Constant	Used by	Meaning	Unit*
<i>crJump</i>	TRVStyle ⁶³⁰ component: <ul style="list-style-type: none"> ▪ TRVStyle.JumpCursor⁶⁴³; ▪ TFontInfo.JumpCursor⁷⁰⁴. 	Hypertext	RVStyle.pas
<i>crRVZoomIn</i>	TRVPrintPreview ⁶²⁵ component (ZoomInCursor ⁸³⁹)	Zoom in	CRVPP.pas
<i>crRVZoomOut</i>	TRVPrintPreview ⁶²⁵ component (ZoomOutCursor ⁸⁴⁰)	Zoom out	CRVPP.pas
<i>crRVSelectRow</i>	TRVTableItemInfo ⁸⁴⁶	Table row selection	RVTable.pas
<i>crRVSelectCol</i>	TRVTableItemInfo ⁸⁴⁶	Table column selection	RVTable.pas
<i>crRVFlipArrow</i>	TRVStyle ⁶³⁰ component (LineSelectCursor ⁶⁴³)	Line selection	RVStyle.pas
<i>crRVLayingIBeam</i>	TRichViewEdit ⁴⁶¹	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
 *	<i>crJump</i>	101
	<i>crRVZoomIn</i>	102
	<i>crRVZoomOut</i>	103
	<i>crRVSelectRow</i>	104
	<i>crRVSelectCol</i>	105

Image	Constant	Value
	<i>crRVFlipArrow</i>	106
	<i>crRVLayingIBeam</i>	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
<i>crJump</i>	<i>crHandPoint</i>
<i>crRVZoomIn</i>	<i>crArrow</i>
<i>crRVZoomOut</i>	<i>crArrow</i>
<i>crRVSelectRow</i>	<i>crUpArrow</i>
<i>crRVSelectCol</i>	<i>crUpArrow</i>
<i>crRVFlipArrow</i>	<i>crArrow</i>
<i>crRVLayingIBeam</i>	<i>crIBeam</i>

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⁽¹²⁴⁾.

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 ⁽¹⁶¹⁾ ; <Item Type> is an index in the collection of styles (RVStyle.TextStyles ⁽⁶⁵⁴⁾)
-1,-3..-6, -10..-12, -60, ...	<i>rvs***</i> ⁽¹⁰⁵⁹⁾ constants	non-text item ⁽¹⁵⁸⁾
-2	<i>rvsCheckpoint</i>	invisible label ("checkpoint" ⁽⁸⁷⁾)
-7	<i>rvsBack</i>	background (image, color)
-8	<i>rvsVersionInfo</i>	RVF version (only in <Version Info>)
-9	<i>rvsDocProperty</i>	text, paragraph, list styles, layout information, DocProperties ⁽²³¹⁾ , DocParameters ⁽²³⁰⁾

<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⁽⁶⁴⁸⁾) (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*⁽⁹¹⁾.

<Query> can be a digit in the range 0..3

<Query>	Meaning
0	The item content is saved in a text format (<i>rvfoSaveBinary</i> is excluded from RichView.RVFOptions ⁽²⁴⁷⁾)
1	The item is requested from program on loading, see below.
2	The item content is saved in a binary format (<i>rvfoSaveBinary</i> is in RichView.RVFOptions ⁽²⁴⁷⁾).
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⁽³⁹³⁾, OnRVFControlNeeded⁽³⁹²⁾.

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⁽²⁴⁷⁾ (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(1032);
  Tag: TRVTag(1029); 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⁽³⁷²⁾ 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⁽¹⁶¹⁾ (Items)

<Record Type> >= 0

<Rest of Header> := [<Additional Properties>] [<Hint>]

<Hint> := <quoted string>, value of rvsPHint⁽¹⁰⁰⁵⁾ 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⁽¹⁶¹⁾.

Break⁽¹⁶⁷⁾ (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=*c/None* 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⁽⁹⁹⁵⁾ value converted to integer.

<Break Width> is a width of line.

<String Count> >= 0.

Picture⁽¹⁶³⁾ (Item)

<Record Type> = *rvsPicture*

<String Count> >= 3

<Rest of Header> = <Vertical Align>

<Vertical Align> ::= <int>

<Vertical Align> is a vertical alignment of picture (TRVVAlign¹⁰³³ value converted to integer)

Strings after the header:

- <Picture Name>
- <Picture Delphi Type>
- <Picture Data>

<Picture Name> ::= <special string 2>, it is an item text.

<Picture Delphi Type> is a name of picture class (such as 'TBitmap', 'TIcon', 'TMetafile', 'TJPEGImage').

<Picture Data> can be in text or binary format.

In text format, each byte is represented as two hexadecimal digits, so all data is encoded in one string.

In binary format, <CrLf> after <Picture Delphi Type> must be CR+LF (both characters). Binary data consists of <Data Size> and <Data>. Data Size is a binary 4-byte number, the number of bytes in <Data>.

Note 1: [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 RVFOptions²⁴⁷ property. If *rvfolgnoreUnknownPicFmt* 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 RVFWarnings²⁵¹ property will contain *rvfwUnknownPicFmt*.

Control¹⁶⁸ (Item)

<Record Type> = *rvcComponent*

See the previous section, adjusted for: *rvfolgnoreUnknownCtrls*, *rvfwUnknownCtrls*.

Bullet¹⁷⁰ (Item)

<Record Type> = *rvcBullet*

<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⁽³⁹³⁾ 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⁽²⁴⁷⁾ property. If *rvfoConvLargeImageIdxToZero* option is defined then index is set to 0. Otherwise the function returns False (reading failed). In both cases RVFWarnings⁽²⁵¹⁾ property will contain *rvfwConvLargeImageIdx*.

Hotspot⁽¹⁷²⁾ (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⁽²³¹⁾

<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 ⁽²⁵³⁾ .TextStyles ⁽⁶⁵⁴⁾)
2	Collection of paragraph styles (Style ⁽²⁵³⁾ .ParaStyles ⁽⁶⁴⁸⁾)
3	Layout (margins, BiDiMode ⁽²²⁴⁾ , MaxTextWidth ⁽²³⁹⁾ , MinTextWidth ⁽²⁴⁰⁾) of TRichView
4	Collection of list styles (Style ⁽²⁵³⁾ .ListStyles ⁽⁶⁴⁶⁾)
5	DocProperties ⁽²³¹⁾
6	Special information, written by TRVReportHelper ⁽⁸⁰⁴⁾ when saving page

7	DocParameters ⁽²³⁰⁾
8	Subdocument (headers and footers for TRichViewEdit)
9	RVStyle.Units ⁽⁶⁵⁵⁾
10	Collection of style templates (Style ⁽²⁵³⁾ .StyleTemplates ⁽⁶⁵²⁾)
11	List of names of style templates
12	DocObjects ⁽²²⁹⁾
13	GetReadDocumentPixelsPerInch ⁽³¹¹⁾ 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⁽⁷³⁵⁾ (<special string>), the second line contains its id⁽⁷³⁴⁾.

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

<Record Type> = *rvsBackground*

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>

<Background Style>::= <int> (TRVBackgroundStyle⁽²²³⁾ value converted to integer)

<Color>::=<int> (background color)

If background style is *bsNoBitmap* (or background bitmap is empty), there are no strings after header (and <Query>=0).

Strings after the header:

- <Picture Delphi Type>
- <Picture Data>

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).

List Markers⁽¹⁷⁴⁾ (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⁽¹⁷⁶⁾ (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⁽⁹¹⁵⁾ property. <Text Style> ::= <int>, it is TextStyleNo⁽⁹¹⁵⁾ property. <Min Width> ::= <int>, it is MinWidth⁽⁹¹⁴⁾ property. <Alignment> ::= <int>, it is ord(Alignment⁽⁹¹⁴⁾). The next line depends on ProtectTextStyleNo⁽⁹¹⁴⁾. <Item Name> ::= <string>.

Sequence⁽¹⁷⁷⁾ (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⁽⁹²⁴⁾ property. <Numbering Type> ::= <int>, it is NumberType⁽⁹²³⁾ property. <Reset> ::= <int>, it is ord(Reset⁽⁹²³⁾). <Start From> ::= <int>, it is Reset⁽⁹²⁴⁾ property. <Text Style>. The next five lines have the same meaning as for labels. <Formatting String> ::= <special string>, it is FormatString⁽⁹²³⁾ property.

Footnote⁽¹⁸⁰⁾ and EndNote⁽¹⁷⁸⁾ (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⁽⁹²⁶⁾ 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⁽¹⁸⁴⁾ (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⁽¹⁰⁰⁰⁾ and string⁽¹⁰⁰⁵⁾ 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⁽²³¹⁾, extra string properties⁽¹⁰⁰⁵⁾, tag strings (if *rvTagsArePChars* in Options⁽²⁴⁰⁾; note: since TRichView v13.2 tags are always strings), style names (if *rvfoUseStyleNames* in RVFOptions⁽²⁴⁷⁾) 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.

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)

TRVOfficeConverter⁷⁹⁵ 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.CachePageImage = True`.

Strings

UTF-16 strings

Most methods and properties in the components have `TRVUnicodeString` parameters, which is defined as `UnicodeString` in Lazarus. These string contain Unicode (UTF-16) text.

ANSI strings

Parameters in `***A` methods have `TRVAnsiString` parameters, which is defined as `AnsiString` in Lazarus. They are treated as ANSI string (not UTF-8!).

These methods include:

- in `TRichView`: `AddNLA`, `AddTextNLA`, `GetItemTextA`, `GetSelTextA`, `GetWordAtA`, `SearchTextA`, `SetItemTextA`
- in `TRichViewEdit`: `GetCurrentItemTextA`, `InsertStringATag`, `InsertTextA`, `SearchTextA`, `SetCurrentItemTextA`, `SetItemTextEdA`

UTF-8 strings

Some methods have `String` parameters. They are treated as UTF-8 in Lazarus

These methods include:

- in `TRichView`: `AddFmt`, `AddNL`, `AddTextNL`, `GetItemText`, `GetSelText`, `GetWordAt`, `SearchText`, `SetItemText`
- in `TRichViewEdit`: `GetCurrentItemText`, `InsertStringTag`, `InsertText`, `SearchText`, `SetCurrentItemText`, `SetItemTextEd`

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\Comboltem\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 ⁽¹⁵⁴⁾ 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 ⁽⁹⁹⁶⁾ type
Graphic type	Classes inherited from TGraphic: TBitmap, TPngImage, etc.	TRVGraphicFM ⁽⁹⁷⁰⁾ Not supported: Windows Metafiles and GIF animations.	You can use TRVGraphic ⁽⁹⁷⁰⁾ class
Font size type	Integer	Single	You can use TRVFontSize ⁽¹⁰⁰⁹⁾ type
Coordinate type	Integer	Single	You can use TRVCoord, TRVCoordRect, TRVCoordPoint types ⁽⁹⁹⁸⁾
Windows system colors	Can be used (for example, <i>clWindow</i>)	Cannot be used. When reading files, these colors are mapped to the most appropriate TAlphaColor constant (<i>clNone</i> is mapped to <i>TAlphaColorRec.Null</i>)	
Text attributes	All	Not supported: character scaling, character spacing	
Paragraph attributes	All	Not supported: "distribute" alignment	

Feature	VCL and LCL	FireMonkey	Comments
Mouse cursors	Custom cursors are used	Only standard mouse cursors are used	See about cursors ⁽¹⁴²⁾ .
Zooming in TRichView ⁽²¹⁰⁾	DocumentPixelsPerInch ⁽²³³⁾	ZoomPercent ⁽²⁵⁸⁾	

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 ⁽⁹⁷⁰⁾ class.

You can use RVGraphicHandler ⁽¹⁰⁵⁷⁾.GetFileDialogFilter ⁽⁹⁷²⁾ 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 ⁽⁷⁷⁸⁾ and TRVReportHelper.SavePDF ⁽⁸¹⁰⁾ to create PDF files.

4 Overview of Items

Item types

- RichView Item Types ⁽¹⁵⁸⁾
- Text Items ⁽¹⁶¹⁾
- Tabulators ⁽¹⁶²⁾
- Pictures ⁽¹⁶³⁾
- *Hot-Pictures* ⁽¹⁶⁶⁾
- *Breaks* ⁽¹⁶⁷⁾
- Controls ⁽¹⁶⁸⁾
- *Bullets* ⁽¹⁷⁰⁾ (images from image-lists)
- *Hotspots* ⁽¹⁷²⁾
- Tables ⁽¹⁷³⁾
- List markers ⁽¹⁷⁴⁾ (paragraph bullets & numbering)
- Labels ⁽¹⁷⁶⁾ (non-text item looking like text)
- Numbered sequences ⁽¹⁷⁷⁾
- Endnotes ⁽¹⁷⁸⁾
- Footnotes ⁽¹⁸⁰⁾
- Sidenotes ⁽¹⁸¹⁾ (notes displayed in floating boxes)
- Text Boxes ⁽¹⁸³⁾ (floating boxes)
- References to parent notes ⁽¹⁸⁴⁾
- "Page number" fields ⁽¹⁸⁵⁾
- "Page count" fields ⁽¹⁸⁶⁾
- Custom Item Types ⁽¹⁸⁹⁾

4.1 Overview

Items

TRichView documents consist of items. Items are indexed from 0 to ItemCount ⁽²³⁷⁾ -1. Some items may contain sub-documents which in their order contain items. Such items are:

- tables ⁽¹⁹⁰⁾ (each cell is a sub-document displayed in the same TRichView control)
- notes (endnotes ⁽¹⁷⁸⁾, footnotes ⁽¹⁸⁰⁾, sidenotes ⁽¹⁸¹⁾) and text boxes ⁽¹⁸³⁾ (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, they are not included in the item range from 0 to ItemCount ⁽²³⁷⁾ -1.

Item Types

A type (also called "style" in this manual) identifier of the given item is returned by the method GetItemStyle ⁽³⁰²⁾ (in editor, a style of the item at the position of caret is returned by CurItemStyle ⁽⁴⁷³⁾).

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 ⁽⁶⁵⁴⁾ collection of the linked ⁽²⁵³⁾ TRVStyle ⁽⁶³⁰⁾ component.

If the returned value is negative, this is a constant identifying the item class.

The standard TRichView item types are:

- text¹⁶¹
- tabulators¹⁶²
- pictures¹⁶³
- *hot-pictures*¹⁶⁶ (pictures with hyperlink)
- *breaks*¹⁶⁷ (horizontal lines)
- controls¹⁶⁸ (any Delphi/C++Builder/Lazarus control)
- *bullets*¹⁷⁰ (images from image-list)
- *hotspots*¹⁷² (bullets with hyperlink)
- tables¹⁷³
- list markers¹⁷⁴ (paragraph bullets & numbering)
- labels¹⁷⁶ (non-text item looking like text)
- numbered sequences¹⁷⁷
- endnotes¹⁷⁸
- footnotes¹⁸⁰
- sidenotes¹⁸¹
- references to parent notes¹⁸⁴
- text boxes¹⁸³
- mathematical expressions¹⁸⁷

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 Unicode¹³⁰ encoding. ANSI text must not contain characters (CR (#13), LF (#10), #0, TAB (#9), FF (12)).

See GetItemText³⁰³, SetItemText³⁶⁰ 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 TRichView⁹¹

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 TRichView⁸⁷

Pagebreak-before Flag

If this flag is set, this item starts a new page when printing.

See `PageBreaksBeforeItems` ⁽²⁴⁴⁾.

Paragraph Attributes

Items are organized in paragraphs. Paragraph attributes are defined in the `ParaStyles` ⁽⁶⁴⁸⁾ collection of the linked `TRVStyle` ⁽⁶³⁰⁾ component.

`GetItemPara` ⁽³⁰¹⁾ returns an index of paragraph style for the given item. This method returns the same value for all items in the same paragraph.

`IsParaStart` ⁽³¹⁸⁾ returns True, if the given item starts a new paragraph.

`IsFromNewLine` ⁽³¹⁷⁾ 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` ⁽¹⁰³³⁾).

You can move item up or down by the specified number of pixels/twips/EMU or percent. See the following extra item properties ⁽¹⁰⁰⁰⁾: `rvepVShift`, `rvepVShiftAbs`.

For text items, see `VShift` ⁽⁷¹⁰⁾ property of text style.

Spacing

You can define spacing around the item. See the following extra item properties ⁽¹⁰⁰⁰⁾: `rvepSpacing`

Coordinates

`GetItemCoords` ⁽²⁹⁸⁾ returns coordinates of the top left corner of the given item relative to the beginning of the document.

`GetItemClientCoords` ⁽²⁹⁸⁾ 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` ⁽⁷⁰⁵⁾ properties of text style.

You can protect non-text items from deletion. See the following extra item integer properties ⁽¹⁰⁰⁰⁾: `rvepDeleteProtect`.

Visibility

Items can be hidden ⁽¹⁰¹⁾. Hidden items are displayed only if `rvoShowHiddenText` is included in the `Options` ⁽²⁴⁰⁾ property of `TRichView` ⁽²¹⁰⁾ control.

Text items are hidden, if `rvteHidden` is included in the `Options` ⁽⁷⁰⁴⁾ property of their style ⁽⁷¹²⁾.

Non-text items are hidden, if a non-zero value is assigned to *rvepHidden* extra item integer property⁽¹⁰⁰⁰⁾.

Popup Hints

The component can show a hint message when the mouse pointer is above the item.

See the following extra item string properties⁽¹⁰⁰⁵⁾: *rvespHint*.

Hint message can be dynamically customized in *OnItemHint*⁽³⁸¹⁾ event.

Events

OnItemAction⁽³⁸⁰⁾ occurs when item is inserted, deleted, moved to undo list, or when item text is modified.

OnSaveItemToFile⁽⁴⁰⁴⁾ 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⁽¹³⁰⁾ encoding.

Style⁽³⁰²⁾ of this item type: ≥ 0 (index in the *TextStyles*⁽⁶⁵⁴⁾ collection of the linked⁽²⁵³⁾ *TRVStyle*⁽⁶³⁰⁾).

Item of this collection (*TFontInfo*⁽⁷¹²⁾ object) completely defines all properties of this text.

Main properties of *TFontInfo*⁽⁷¹²⁾:

- *Jump*⁽⁷¹⁴⁾ - determines if this text is hypertext⁽⁹²⁾ or not;
- properties defining font, colors, background, hypertext cursors, protection, bi-di mode⁽¹³²⁾, and others.

Text item cannot contain:

- CR and LF (#13 and #10) characters;
- TAB (#09) characters (they are inserted as tabulators⁽¹⁶²⁾ or replaced with several space characters, depending on *SpacesInTab*⁽⁶⁵²⁾);
- #0 characters;
- FF (#12) characters (see *PageBreaksBeforeItems*⁽²⁴⁴⁾ property of *TCustomRichView*).

Methods of *TCustomRichView*

⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- single item: *AddNL*⁽²⁷⁰⁾ (and shortcut *Add*⁽²⁷⁰⁾), *AddFmt*⁽²⁶⁶⁾
- multiple items: *AddTextNL*⁽²⁷⁴⁾.

The following viewer-style methods change text of the given item

- *SetItemText* -A -W⁽³⁶⁰⁾

The following methods return text of the given item:

- *GetItemText* -A -W⁽³⁰³⁾

Appending text from file/stream:

- LoadText, LoadTextW⁽³³⁰⁾,
- LoadTextFromStream, LoadTextFromStreamW⁽³³¹⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style methods insert item of this type at the position of caret:

- single item: InsertStringTag, InsertStringATag, InsertStringWTag⁽⁵³⁰⁾;
- multiple items: InsertText, -A, -W⁽⁵³²⁾.

The following editor-style methods change text of the given item:

- SetItemTextEd -A -W⁽⁵⁶⁴⁾, SetCurrentItemText -A -W⁽⁵⁵⁷⁾

The following methods return text of item at the position of caret:

- GetCurrentItemText -A -W⁽⁵¹¹⁾

The following editor-style methods insert text from files at the position of caret:

- InsertTextFromFile, InsertTextFromFileW⁽⁵³³⁾

See Also...

See also methods for copying and pasting⁽¹⁰⁸⁾, selecting text⁽¹⁰⁷⁾.

Some of the methods above can insert tabulators⁽¹⁶²⁾ as well.

See also: classes for item types⁽⁸⁴⁴⁾.

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⁽²³⁸⁾ and defined by the following properties:

- paragraph's LeftIndent⁽⁷²²⁾ (for paragraph with hanging (negative) FirstIndent⁽⁷²²⁾);
- paragraph's collection of Tabs⁽⁷²⁶⁾
- TRVStyle.DefTabWidth⁽⁶³⁶⁾.

Style⁽³⁰²⁾ of this item type: *rvsTab* (-12)

Properties of this item type are defined by:

- its text style (the same properties as for text items⁽¹⁶¹⁾)
- corresponding tab stop (item in the paragraph's collection of Tabs⁽⁷²⁶⁾: TRVTabInfo⁽⁷⁴⁶⁾)

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⁽⁶⁵²⁾ property of the linked TRVStyle⁽⁶³⁰⁾ component must be 0 (default). If it is positive, several spaces are inserted instead of tabulator.

Editor accepts **Tab** keys only if *rvsWantTabs* is included in EditorOption⁽⁴⁷⁵⁾ property.

Tabulators are read from RVF (RichView Format) files and streams regardless of value of SpacesInTab⁽⁶⁵²⁾ 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`⁽²⁴⁰⁾.

Methods of `TCustomRichView`⁽²¹⁰⁾

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`⁽²⁷⁴⁾;
- `LoadText`, `LoadTextW`⁽³³⁰⁾, `LoadTextFromStream`, `LoadTextFromStreamW`⁽³³¹⁾;
- `AddTab`⁽²⁷³⁾ – adds tabulator regardless of `SpacesInTab` mode.

Methods of `TCustomRichViewEdit`⁽⁴⁶¹⁾

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`⁽⁵³²⁾, `InsertTextFromFile`, `InsertTextFromFileW`⁽⁵³³⁾
- `InsertTab`⁽⁵³¹⁾ – inserts tabulator regardless of `SpacesInTab` mode

See also methods for pasting⁽¹⁰⁸⁾.

4.4 Pictures

Pictures

Picture item displays a picture stored in `TRVGraphic`⁽⁹⁷⁰⁾ 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`⁽¹⁵⁷⁾, stores WebP and WBMP)
- `TSvgGraphic` (`Skia4Delphi`⁽¹⁵⁷⁾, stores SVG; include `RVSkia` unit in your project)

In addition, you can use third-party graphic classes⁽¹⁰⁶⁸⁾.

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`⁽¹⁵⁷⁾) (wrapper for `TSvgBrush`, stores SVG, include `fmRVSkiaFM` in your project)

Style⁽³⁰²⁾ of this item type: *rsvPicture* (-3)

For hypertext pictures, see *hot-pictures*⁽¹⁶⁶⁾ item type.

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddPicture⁽²⁷²⁾,

The following viewer-style method changes the main properties of item of this type:

- SetPictureInfo⁽³⁶³⁾

The following method returns main properties of the given item of this type:

- GetPictureInfo⁽³¹⁰⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertPicture⁽⁵²⁶⁾

The following editor-style methods modify main properties of the item of this type:

- SetCurrentPictureInfo⁽⁵⁵⁹⁾,
- SetPictureInfoEd⁽⁵⁶⁵⁾

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentPictureInfo⁽⁵¹³⁾

Properties

Size

This item type has the following integer properties⁽¹⁰⁰⁰⁾ related to size:

- *rvepImageWidth*, *rvepImageHeight* stretch the image.

Layout and appearance

This item type has the following integer properties⁽¹⁰⁰⁰⁾ 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⁽¹⁰⁰⁰⁾ is nonzero, TRichView does not free it.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see *RichViewMaxPictureCount*⁽¹⁰⁴⁶⁾). Shared images are never deactivated.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign⁽¹⁰³³⁾;

- extra integer properties⁽¹⁰⁰⁰⁾: *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*⁽⁴⁷⁵⁾. You can forbid resizing for the specific image by assigning 0 to *rvepResizable* integer property⁽¹⁰⁰⁰⁾.

Resizing with mouse does not modify the image itself, it changes *rvepImageWidth* and *rvepImageHeight*.

OnItemResize⁽⁵⁷⁷⁾ event occurs after resizing.

Other properties

This item type has the following specific integer properties⁽¹⁰⁰⁰⁾:

- *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, and are not saved in HTML for this image.

This item type has the following specific string properties⁽¹⁰⁰⁵⁾:

- *rvespAlt* – text representation of the image, saved in HTML as attribute;
- *rvespImageFileName* – this property can be saved in HTML file as image location.

Saving and Loading [VCL and LCL]

RichView Format (RVF)⁽¹²⁴⁾

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*⁽⁹⁷²⁾.

If *rvfoSavePicturesBody* is excluded from *RVFOptions*⁽²⁴⁷⁾, graphic data is not saved in RVF. When loading such RVF files, *OnRVFPictureNeeded*⁽³⁹³⁾ event occurs. You can use this feature to store images in database or files.

Export to HTML⁽¹²⁷⁾

By default, all pictures are saved as JPEGs. If you want to save pictures of some graphic class without conversion, call *RVGraphicHandler*⁽¹⁰⁵⁷⁾.`RegisterHTMLGraphicFormat`.

For example:

- Delphi: `RVGraphicHandler(1057).RegisterHTMLGraphicFormat(TMyGifImage);`
- C++Builder: `RVGraphicHandler(1057).RegisterHTMLGraphicFormat(__classid(TMyGifImage));`

The standard PNG and GIF formats may be automatically registered by TRichView, see the topic about *RVGraphicHandler*⁽⁹⁷²⁾.

If you want to save pictures in another format, use `OnHTMLSaveImage`³⁷⁵ or `OnSaveImage2`⁴⁰² events.

RTF export

`RTFOptions`²⁴⁵ 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`¹⁰⁵⁷.`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`¹⁰⁵⁷.`RegisterPngGraphic`;
- icons (they are saved as PNG in Delphi 2009+ using `TPngImage` class)

RTF and DocX Import

You can set `RTFReadProperties`²⁴⁶.`IgnorePictures`⁴⁵⁵ 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 classes¹⁰⁶⁸.

See Also...

- Working in 256-color display mode²³⁴;
- Animation¹¹⁹;
- Smooth image scaling¹²⁰.

4.5 Hot-pictures

"Hot-picture" is a picture¹⁶³ - hypertext link⁹².

Style³⁰² of this item type: `rvsHotPicture` (-10)

All methods working with pictures¹⁶³ work with hot-pictures. The difference is only in methods that insert hot-picture in the document.

Methods of TCustomRichView ⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddHotPicture ⁽²⁶⁷⁾.

Methods of TCustomRichViewEdit ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertHotPicture ⁽⁵¹⁹⁾

ConvertToHotPicture ⁽⁵⁰¹⁾, ConvertToPicture ⁽⁵⁰²⁾ 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 ⁽³⁰²⁾ of this item type: *rvsBreak* (-1)

Methods of TCustomRichView ⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddBreak ⁽²⁶²⁾
- The following viewer-style method changes the main properties of item of this type:
- SetBreakInfo ⁽³⁵¹⁾

The following method returns main properties of the given item of this type:

- GetBreakInfo ⁽²⁸⁹⁾

Methods of TCustomRichViewEdit ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertBreak ⁽⁵¹⁵⁾

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBreakInfo ⁽⁵⁵¹⁾,
- SetBreakInfoEd ⁽⁵⁴⁷⁾

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentBreakInfo ⁽⁵⁰⁴⁾

Properties

Main properties of *breaks*: style, width, color.

In addition to the methods listed above, they are accessible as *rveipcBreakStyle*, *rveipcBreakWidth*, *rveipcBreakColor* ⁽¹⁰⁵¹⁾ properties.

Saving and loading

Export to HTML ⁽¹²⁷⁾

Breaks are exported as `<hr>`. Color of *break* is retained only in CSS version (HTMLSaveProperties⁽²³⁷⁾.HTMLSavingType⁽⁴³³⁾ = *rvhtmlstNormal*, or *rvcsssoForceNonTextCSS* is included in HTMLSaveProperties⁽²³⁷⁾.CSSOptions⁽⁴³⁰⁾).

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⁽³⁴⁵⁾), 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⁽²³⁹⁾.DefaultItemProperties⁽⁴⁴⁴⁾.BreakColor⁽⁴³⁸⁾.

4.7 Controls

You can insert any Delphi/C++Builder control in the document.

Style⁽³⁰²⁾ of this item type: *rvsComponent* (-5)

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddControl⁽²⁶⁵⁾.

The following viewer-style method changes the main properties of item of this type:

- SetControllInfo⁽³⁵⁴⁾

The following method returns main properties of the given item of this type:

- GetControllInfo⁽²⁹³⁾

See also

- FindControllItemNo⁽²⁸⁷⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertControl⁽⁵¹⁷⁾

The following editor-style methods modify main properties of the item of this type:

- SetCurrentControllInfo⁽⁵⁵⁴⁾,
- SetControllInfoEd⁽⁵⁵⁰⁾

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentControllInfo⁽⁵⁰⁶⁾

Also

- `ResizeControl` ⁽⁵⁴²⁾,
- `ResizeCurrentControl` ⁽⁵⁴³⁾,
- `AdjustControlPlacement2` ⁽⁴⁸⁹⁾

Events:

- `OnControlAction` ⁽³⁷³⁾ – occurs when the control is loaded from RVF ⁽¹²⁴⁾, deleted, moved to/from undo buffer, saved to RVF.

Properties

Layout and appearance

This item type has the following integer properties ⁽¹⁰⁰⁰⁾ 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 ⁽¹⁰⁵¹⁾ 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 ⁽¹⁰⁰⁰⁾ is nonzero, `TRichView` does not free it.

Vertical Position

This item type has the following properties affecting vertical position:

- `VAlign` ⁽¹⁰³³⁾;
- extra integer properties ⁽¹⁰⁰⁰⁾: `rvepVShift`, `rvepVShiftAbs`.

Resizing

You can allow resizing this control in editor by assigning non-zero value to its `rvepResizable` property ⁽¹⁰⁰⁰⁾.

You can forbid mouse resizing of all images and controls in the editor by including `rvoNoImageResize` in `EditorOptions` ⁽⁴⁷⁵⁾.

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` ⁽³⁴⁸⁾.

`OnItemResize` ⁽⁵⁷⁷⁾ event occurs after resizing.

Other properties

This item type has the following specific integer properties ⁽¹⁰⁰⁰⁾:

- `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⁽⁹¹⁾; you can use item's tag as you wish).

FireMonkey: TRichView does not use tags of inserted components.

Saving and Loading

RVF⁽¹²⁴⁾

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⁽²⁴⁷⁾, control itself is not saved in RVF. When loading such RVF files, OnRVFControlNeeded⁽³⁹²⁾ event occurs.

Export to text, RTF, DocX, and HTML⁽¹²⁷⁾

By default, controls are not exported. Use OnSaveComponentToFile⁽³⁹⁷⁾ 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⁽⁸³¹⁾ event.

Special Width Mode

Width of controls can be set in percent:

```
var Item: TRVControlItemInfo(844);
    ...
Item := RichView.GetItem(296)(ItemNo) as TRVControlItemInfo;
Item.PercentWidth := 50; // 50% of document width (minus paragraph indents)
```

The effect will be after reformatting⁽²⁸⁸⁾.

See also...

BeginOleDrag⁽²⁷⁸⁾

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 ⁽¹⁷⁴⁾.

Style ⁽³⁰²⁾ of this item type: *rvsBullet* (-6)

For hypertext *bullets*, see *hotspot* ⁽¹⁷²⁾ item type.

Methods of TCustomRichView ⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddBullet ⁽²⁶³⁾

The following viewer-style method changes the main properties of item of this type:

- SetBulletInfo ⁽³⁵²⁾

The following method returns main properties of the given item of this type:

- GetBulletInfo ⁽²⁸⁹⁾

Methods of TCustomRichViewEdit ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertBullet ⁽⁵¹⁵⁾

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBulletInfo ⁽⁵⁵²⁾,
- SetBulletInfoEd ⁽⁵⁴⁸⁾

The following method returns main properties of the item of this type at the position of caret:

GetCurrentBulletInfo ⁽⁵⁰⁵⁾

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 *rveipclmageIndex* ⁽¹⁰⁵¹⁾ property.

Layout and appearance

This item type has the following integer properties ⁽¹⁰⁰⁰⁾ 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 ⁽¹⁰³³⁾;
- extra integer properties ⁽¹⁰⁰⁰⁾: *rvepVShift*, *rvepVShiftAbs*.

Resizing

Bullets cannot be resized.

Other properties

This item type has the following specific string properties⁽¹⁰⁰⁵⁾:

- *rvespAlt* – text representation of the image, saved in HTML as attribute;

Memory Usage

RichView does not own this image-list, and it does not destroy it when clearing document.

Saving and Loading

RVF⁽¹²⁴⁾

ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded⁽³⁹³⁾ occurs allowing you to provide image-lists for *bullets* and *hotspots*.

Export to RTF, DocX and HTML⁽¹²⁷⁾

Bullets are exported as pictures. If you want to write special code for exporting bullets in HTML, use OnHTMLSaveImage⁽³⁷⁵⁾ event.

4.9 Hotspots

"Hotspot" is a *bullet*⁽¹⁷⁰⁾ (image from TImageList) with hypertext link⁽⁹²⁾.

Hotspots can change their image under the mouse pointer.

Style⁽³⁰²⁾ of this item type: *rvsHotspot* (-4)

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- AddHotspot⁽²⁶⁸⁾.

The following viewer-style method changes the main properties of item of this type:

- SetHotspotInfo⁽³⁵⁷⁾

The following method returns main properties of the given item of this type:

- GetHotspotInfo⁽²⁹⁵⁾,
- GetBulletInfo⁽²⁸⁹⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertHotspot⁽⁵²⁰⁾

The following editor-style methods modify main properties of the item of this type:

- SetCurrentHotspotInfo⁽⁵⁵⁵⁾,
- SetHotspotInfoEd⁽⁵⁶⁰⁾

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentHotspotInfo⁽⁵⁰⁷⁾,
- GetCurrentBulletInfo⁽⁵⁰⁵⁾

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`⁽¹⁰⁵¹⁾ 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⁽¹⁹⁰⁾,
- `TRVTableItemInfo`⁽⁸⁴⁶⁾ (see for detailed information).

Style⁽³⁰²⁾ 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`⁽⁴⁶¹⁾.

Extra Properties

This item type has the following specific string properties⁽¹⁰⁰⁵⁾:

- *vespImageFileName* – value of this property can be saved in HTML file as table background image location; also accessible as `table.BackgroundImageFileName`⁽⁸⁵⁰⁾.

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style methods add item of this type to the end of the document:

- `AddItem`⁽²⁶⁹⁾ (create `TRVTableItemInfo`⁽⁸⁴⁶⁾ object, add it using `AddItem`⁽²⁶⁹⁾)

The following method returns `TRVTableItemInfo`⁽⁸⁴⁶⁾ object

- `GetItem`⁽²⁹⁶⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem`⁽⁵²²⁾ (create `TRVTableItemInfo`⁽⁸⁴⁶⁾.object, insert it using `InsertItem`⁽⁵²²⁾)

The following methods return `TRVTableItemInfo`⁽⁸⁴⁶⁾ object at the position of caret:

- `GetCurrentItem`⁽⁵⁰⁸⁾
- `GetCurrentItemEx`⁽⁵⁰⁹⁾.

Resizing

Users can resize table rows and columns, if *rvtoRowSizing* and/or *rvtoColSizing* are included in *table.Options*⁽⁸⁶¹⁾.

Resizing changes *BestWidth*⁽⁸⁹⁸⁾ and *BestHeight*⁽⁸⁹⁸⁾ properties of table cells.

OnItemResize⁽⁵⁷⁷⁾ event occurs after resizing.

Saving and Loading

See Export of tables⁽²⁰³⁾.

4.11 List markers

This is a very special item type representing paragraph bullet or number.

Style⁽³⁰²⁾ of this item type: *rvsListMarker* (-11)

See also: numbered sequences⁽¹⁷⁷⁾.

Properties of List Markers

TRVStyle⁽⁶³⁰⁾ component has a collection of list styles (*ListStyles*⁽⁶⁴⁶⁾). Each list style⁽⁷³¹⁾ contains a collection of list levels (*Levels*⁽⁷³²⁾ property). List level⁽⁷⁵⁰⁾ 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⁽⁶⁴⁶⁾;
- *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*⁽³⁶²⁾,
- *GetListMarkerInfo*⁽³⁰⁷⁾,
- *RefreshListMarkers*⁽³³²⁾.

Methods of *TCustomRichViewEdit*:

- *ApplyListStyle*⁽⁴⁹⁰⁾,
- *RemoveLists*⁽⁵⁴²⁾,
- *ChangeListLevels*⁽⁵⁰⁰⁾.

Saving and Loading

RVF⁽¹²⁴⁾

For list levels with ImageLists: ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded³⁹³ occurs allowing you to provide an image-list.

HTM Export, HTMLSaveProperties²³⁷.ListMarkerSavingType⁴³⁶ = *rvhtmlmstNormal*

Lists are saved using and HTML tags.

Export using CSS (HTMLSaveProperties²³⁷.HTMLSavingType⁴³³ = *rvhtmlstNormal*):

CSS of lists are partially inserted directly in HTML (as inline attributes), partially are saved in a style sheet.

- *rvlstBullet*, *rvlstUnicodeBullet*⁷⁵⁵ are exported as bullets with default markers (disc, circle or square) if possible. Otherwise, FormatString⁷⁵² is used. Font⁷⁵¹ is ignored;
- *rvlstDecimal*, *rvlstLowerAlpha*⁷⁵⁵ and other numbered list types are exported as lists with corresponding numbering. FormatString⁷⁵² and Font⁷⁵¹ are ignored;
- *rvlstPicture*, *rvlstImageList*, *rvlstImageListCounter*⁷⁵⁵ are exported as bullets with picture. It's possible to change default image saving by OnHTMLSaveImage³⁷⁵.

Indents are saved.

Export without CSS (HTMLSaveProperties²³⁷.HTMLSavingType⁴³³ = *rvhtmlstSimplified*):

- *rvlstBullet*, *rvlstUnicodeBullet*, *rvlstPicture*, *rvlstImageList*⁷⁵⁵ are exported as bullets with default markers (disc, circle or square). FormatString⁷⁵², Picture⁷⁵⁹, ImageList⁷⁵⁴ and Font⁷⁵¹ are ignored;
- *rvlstDecimal*, *rvlstLowerAlpha*⁷⁵⁵ and other numbered list types are exported as lists with corresponding numbering (or as decimal, if this numbering type is not supported in HTML). FormatString⁷⁵² and Font⁷⁵¹ are ignored;
- *rvlstImageListCounter*⁷⁵⁵ is saved as a decimal numbering.

Indents are not saved.

HTML Export, HTMLSaveProperties²³⁷.ListMarkerSavingType⁴³⁶ = *rvhtmlmstAsText*

Markers are saved without using special HTML keywords for lists (like , ,).

Font⁷⁵¹ and format strings⁷⁵² settings are respected.

Saving with CSS (HTMLSaveProperties²³⁷.HTMLSavingType⁴³³ = *rvhtmlstNormal*):

LeftIndent⁷⁵⁵ and MarkerIndent⁷⁵⁷ are retained. FirstIndent⁷⁵¹ are retained for non-hanging markers only (MarkerIndent>=LeftIndent).

Saving without CSS (HTMLSaveProperties²³⁷.HTMLSavingType⁴³³ = *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⁽¹⁶¹⁾. This text cannot be wrapped (always displayed on one line).

Class for this item type is **TRVLabelItemInfo**⁽⁹¹²⁾ (see for detailed information).

Style⁽³⁰²⁾ of this item type: *rvsLabel* (-200)

The following items are inherited from labels:

- numbered sequences⁽¹⁷⁷⁾
- footnotes⁽¹⁸⁰⁾
- endnotes⁽¹⁷⁸⁾
- sidenotes⁽¹⁸¹⁾
- text boxes⁽¹⁸³⁾
- references to notes⁽¹⁸⁴⁾

Related Properties of TRVStyle⁽⁶³⁰⁾

- FieldHighlightColor⁽⁶³⁷⁾ – color for highlighting label items.
- FieldHighlightType⁽⁶³⁸⁾ specifies when to highlight label items.

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- AddItem⁽²⁶⁹⁾ (create TRVLabelItemInfo⁽⁹¹²⁾ object, add it using AddItem⁽²⁶⁹⁾)

The following method returns TRVLabelItemInfo⁽⁹¹²⁾ object

- GetItem⁽²⁹⁶⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertItem⁽⁵²²⁾ (create TRVLabelItemInfo⁽⁹¹²⁾ object, insert it using InsertItem⁽⁵²²⁾)

The following method returns TRVLabelItemInfo⁽⁹¹²⁾ object at the position of caret:

- GetCurrentItem⁽⁵⁰⁸⁾.

The following methods change properties as editing operations:

- SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾;
- SetItemExtraIntPropertyExEd⁽⁵⁶²⁾;
- SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾;
- SetItemExtraStrPropertyExEd⁽⁵⁶²⁾.

TextStyleNo⁽⁹¹⁵⁾ property can be modified by ApplyTextStyle⁽⁴⁹⁴⁾ and ApplyStyleConversion⁽⁴⁹²⁾ methods.

Properties

Main properties

- Text⁽⁹¹⁵⁾ – text to display;
- TextStyleNo⁽⁹¹⁵⁾ defines font and colors. This is an index in TextStyles⁽⁶⁵⁴⁾ collection of the linked TRVStyle⁽⁶³⁰⁾ component;
- ProtectTextStyleNo⁽⁹¹⁴⁾ – if *True*, editing operations cannot change TextStyleNo⁽⁹¹⁵⁾;

- `MinWidth`⁹¹⁴ – minimal width for the label item;
- `Alignment`⁹¹⁴ – horizontal alignment of text (if the item width is increased by `MinWidth`⁹¹⁴);
- `Cursor`⁹¹⁴ – mouse cursor above this item.
- `RemoveInternalLeading`⁹¹⁵ – if *True*, the font internal leading is not included in the item height.

These properties are also accessible as `rvespcText`¹⁰⁵⁶, `rveipcProtectTextStyleNo`, `rveipcMinWidth`, `rveipcAlignment`, `rveipcCursor`, `rveipcRemoveInternalLeading`¹⁰⁵¹ properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties¹⁰⁰⁰ 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`¹⁰³³;
- extra integer properties¹⁰⁰⁰: `rvepVShift`, `rvepVShiftAbs`.

4.13 Numbered sequences

Numbered sequence is a non-text item looking like text¹⁶¹ 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`⁹²¹ (see for detailed information).

Style³⁰² of this item type: `rvsSequence` (-202)

The following items are inherited from numbered sequences:

- footnotes¹⁸⁰
- endnotes¹⁷⁸
- sidenotes¹⁸¹
- text boxes¹⁸³

Methods of `TCustomRichView`²¹⁰

The following viewer-style method adds item of this type to the end of the document:

- `AddItem`²⁶⁹ (create `TRVSeqItemInfo`⁹²¹ object, add it using `AddItem`²⁶⁹)

The following method returns `TRVSeqItemInfo`⁹²¹ object

- `GetItem`²⁹⁶

Methods of `TCustomRichViewEdit`⁴⁶¹

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem`⁵²² (create `TRVSeqItemInfo`⁹²¹ object, insert it using `InsertItem`⁵²²)

The following method returns `TRVSeqItemInfo`⁹²¹ object at the position of caret:

- `GetCurrentItem`⁵⁰⁸.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx`⁵⁵⁶;
- `SetItemExtraIntPropertyExEd`⁵⁶²;
- `SetCurrentItemExtraStrPropertyEx`⁵⁵⁷;
- `SetItemExtraStrPropertyExEd`⁵⁶².

Properties

Numbered sequences have all properties of label items¹⁷⁶, but `Text`⁹¹⁵ is calculated automatically.

Additionally, it has properties:

- `NumberType`⁹²³ – numbering type;
- `SeqName`⁹²⁴ – name of a sequence (this item continues numbering from the previous item having the same name of a sequence);
- `Reset`⁹²³ – if *True*, a numbering is not continued but reset from `StartFrom`⁹²⁴.
- `FormatString`⁹²³ – optional format string; if defined, a text to display is formatted according to this string.

These properties are also accessible as `rvespcSeqName`, `rvespcSeqFormatString`¹⁰⁵⁶, `rveipcSeqStartFrom`, `rveipcSeqReset`, `rveipcSeqNumberType`¹⁰⁵¹ properties. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML¹²⁷ 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`⁹²³).

You can set `RTFReadProperties`²⁴⁶.`IgnoreSequences`⁴⁵⁵ 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`⁶³⁷). If `FormatString`⁹²³ 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`⁹²⁸ (see for detailed information).

Style³⁰² of this item type: `rvsEndnote` (-204)

Methods of `TCustomRichView`²¹⁰

The following viewer-style method adds item of this type to the end of the document:

- `AddItem`²⁶⁹ (create `TRVEndnoteItemInfo`⁹²⁸ object, add it using `AddItem`²⁶⁹)

The following method returns TRVEndnoteItemInfo⁽⁹²⁸⁾ object

- GetItem⁽²⁹⁶⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertItem⁽⁵²²⁾ (create TRVEndnoteItemInfo⁽⁹²⁸⁾ object, insert it using InsertItem⁽⁵²²⁾)

The following method returns TRVEndnoteItemInfo⁽⁹²⁸⁾ object at the position of caret:

- GetCurrentItem⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾;
- SetItemExtraIntPropertyExEd⁽⁵⁶²⁾;
- SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾;
- SetItemExtraStrPropertyExEd⁽⁵⁶²⁾.

Properties

Endnotes have all properties of numbered sequences⁽¹⁷⁷⁾.

Differences:

- NumberType⁽⁹²³⁾ is read-only; it always returns TRVStyle.EndnoteNumbering⁽⁶³⁷⁾;
- SeqName⁽⁹²⁴⁾ = '@endnote@' and must not be changed.

Saving and Loading

Export to HTML⁽¹²⁷⁾

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⁽¹⁷⁷⁾ and notes). Endnotes are saved as hyperlinks to this anchor.

RTF and DocX

Endnotes can be exported in RTF and DocX. If FormatString⁽⁹²³⁾ contains a single character, it can be exported to RTF/DocX. Reset⁽⁹²³⁾ and StartFrom⁽⁹²⁴⁾ properties cannot be exported.

You can set RTFReadProperties⁽²⁴⁶⁾.IgnoreNotes⁽⁴⁵⁵⁾ 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`⁶³⁷). If `FormatString`⁹²³ 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`⁹³⁰ (see for detailed information).

Style³⁰² of this item type: `rvsFootnote` (-203)

Methods of `TCustomRichView`²¹⁰

The following viewer-style method adds item of this type to the end of the document:

- `AddItem`²⁶⁹ (create `TRVFootnoteItemInfo`⁹³⁰ object, add it using `AddItem`²⁶⁹)

The following method returns `TRVFootnoteItemInfo`⁹³⁰ object

- `GetItem`²⁹⁶

Methods of `TCustomRichViewEdit`⁴⁶¹

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem`⁵²² (create `TRVFootnoteItemInfo`⁹³⁰ object, insert it using `InsertItem`⁵²²)

The following method returns `TRVFootnoteItemInfo`⁹³⁰ object at the position of caret:

- `GetCurrentItem`⁵⁰⁸.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx`⁵⁵⁶;
- `SetItemExtraIntPropertyExEd`⁵⁶²;
- `SetCurrentItemExtraStrPropertyEx`⁵⁵⁷;
- `SetItemExtraStrPropertyExEd`⁵⁶².

Properties

Footnotes have all properties of numbered sequences¹⁷⁷.

Differences:

- `NumberType`⁹²³ is read-only; it always returns `TRVStyle.FootnoteNumbering`⁶³⁷;
- `SeqName`⁹²⁴ = '@footnote@' and must not be changed.

Saving and Loading

Export to HTML¹²⁷

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 sequences¹⁷⁷ and notes). Footnotes are saved as hyperlinks to this anchor.

RTF and DocX

Footnotes can be exported in RTF and DocX. If `FormatString`⁽⁹²³⁾ contains a single character, it can be exported to RTF/DocX.

You can set `RTFReadProperties`⁽²⁴⁶⁾.`IgnoreNotes`⁽⁴⁵⁵⁾ 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 pla
```

4.16 Sidenotes

A sidenote looks like a number (numbering type is specified in `TRVStyle.SidenoteNumbering`⁽⁶³⁷⁾). If `FormatString`⁽⁹²³⁾ 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`⁽⁹³³⁾ (see for detailed information).

Style⁽³⁰²⁾ of this item type: `rvsSidenote` (-206)

The following items are inherited from numbered sidenotes:

- text boxes⁽¹⁸³⁾

Methods of `TCustomRichView`⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- `AddItem`⁽²⁶⁹⁾ (create `TRVSidenoteItemInfo`⁽⁹³³⁾ object, add it using `AddItem`⁽²⁶⁹⁾)

The following method returns `TRVSidenoteItemInfo`⁽⁹³³⁾ object

- `GetItem`⁽²⁹⁶⁾

Methods of `TCustomRichViewEdit`⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem`⁽⁵²²⁾ (create `TRVSidenoteItemInfo`⁽⁹³³⁾ object, insert it using `InsertItem`⁽⁵²²⁾)

The following method returns `TRVSidenoteItemInfo`⁽⁹³³⁾ object at the position of caret:

- `GetCurrentItem`⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾;
- `SetItemExtraIntPropertyExEd`⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾;
- `SetItemExtraStrPropertyExEd`⁽⁵⁶²⁾.

Properties

Sidenotes have all properties of numbered sequences ⁽¹⁷⁷⁾.

Differences:

- `NumberType` ⁽⁹²³⁾ is read-only; it always returns `TRVStyle.SidenoteNumbering` ⁽⁶³⁷⁾;
- `SeqName` ⁽⁹²⁴⁾ = '@sidenote@' and must not be changed.

Additionally, it has the following properties:

- `BoxPosition` ⁽⁹⁴⁰⁾ – properties relative to the position of a text box;
- `BoxProperties` ⁽⁹³⁷⁾ – 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 ⁽¹⁰⁵¹⁾. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML ⁽¹²⁷⁾

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 ⁽¹⁷⁷⁾ 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` ⁽⁹²⁷⁾ followed by a text box containing `Document` ⁽⁹²⁶⁾. A bookmark named '_sidenoteN' (where *N* is an index of this sidenote) is added around `NoteText`, to allow saving note references ⁽¹⁸⁴⁾ 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`⁽⁹⁴⁰⁾.`VerticalAnchor`⁽⁹⁴⁵⁾;
- MS Word does not support different padding for border and background, so `BoxProperties`⁽⁹³⁷⁾.`Border`⁽⁹³⁸⁾.`BorderOffsets`⁽⁷⁴³⁾ 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⁽¹⁸³⁾.

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 pla
```

4.17 Text boxes

A text box item is a simplified version of a sidenote⁽¹⁸¹⁾. There is no visible text at the place of insertion of this item.

If `rvoShowSpecialCharacters` is included in `RichView.Options`⁽²⁴⁰⁾, 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`⁽⁹⁴⁷⁾ (see for detailed information).

Style⁽³⁰²⁾ of this item type: `rvsTextBox` (-207)

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- `AddItem`⁽²⁶⁹⁾ (create `TRVSidenoteItemInfo`⁽⁹³³⁾ object, add it using `AddItem`⁽²⁶⁹⁾)

The following method returns `TRVSidenoteItemInfo`⁽⁹³³⁾ object

- `GetItem`⁽²⁹⁶⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem`⁽⁵²²⁾ (create `TRVSidenoteItemInfo`⁽⁹³³⁾ object, insert it using `InsertItem`⁽⁵²²⁾)

The following method returns `TRVSidenoteItemInfo`⁽⁹³³⁾ object at the position of caret:

- `GetCurrentItem`⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾;
- `SetItemExtraIntPropertyExEd`⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾;
- `SetItemExtraStrPropertyExEd`⁽⁵⁶²⁾.

Properties

Text box items have all properties of sidenotes ⁽¹⁸¹⁾, but almost all of them are ignored. Only floating box-related properties are used: `BoxPosition` ⁽⁹⁴⁰⁾ and `BoxProperties` ⁽⁹³⁷⁾.

Saving and Loading

Export to HTML ⁽¹²⁷⁾

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 ⁽¹⁸¹⁾ 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 floati

4.18 References to notes

When inserted in footnote ⁽¹⁸⁰⁾'s/endnote ⁽¹⁷⁸⁾'s/sidenote ⁽¹⁸¹⁾'s Document ⁽⁹²⁶⁾, the note reference displays the same text as the parent note.

When inserted in `TCustomRichView`, it displays `NoteText` ⁽²⁴⁰⁾.

Class for this item type is `TRVNoteReferenceItemInfo` ⁽⁹⁴⁹⁾ (see for detailed information).

Style ⁽³⁰²⁾ of this item type: `rvsNoteReference` (-205)

Methods of `TCustomRichView` ⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- `AddItem` ⁽²⁶⁹⁾ (create `TRVNoteReferenceItemInfo` ⁽⁹⁴⁹⁾ object, add it using `AddItem` ⁽²⁶⁹⁾)

The following method returns `TRVNoteReferenceItemInfo` ⁽⁹⁴⁹⁾ object

- `GetItem` ⁽²⁹⁶⁾

Methods of `TCustomRichViewEdit` ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem` ⁽⁵²²⁾ (create `TRVNoteReferenceItemInfo` ⁽⁹⁴⁹⁾ object, insert it using `InsertItem` ⁽⁵²²⁾)

The following method returns `TRVNoteReferenceItemInfo` ⁽⁹⁴⁹⁾ object at the position of caret:

- `GetCurrentItem` ⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx` ⁽⁵⁵⁶⁾;
- `SetItemExtraIntPropertyExEd` ⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrPropertyEx` ⁽⁵⁵⁷⁾;
- `SetItemExtraStrPropertyExEd` ⁽⁵⁶²⁾.

Saving and Loading

Export to HTML ⁽¹²⁷⁾

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.IgnoreNotes` ⁽²⁴⁶⁾ to `True`, and note references will not be imported from RTF/DocX.

4.19 Page numbers

When inserted in `TRichView` ⁽²¹⁰⁾ or `TRichViewEdit` ⁽⁴⁶¹⁾, this item is displayed as '{p}'.

When printing with `TRVPrint` ⁽⁷⁵⁹⁾, this item displays a page number where this item is located. Page numbers are counted from `TRVPrint.PageNoFromNumber` ⁽⁸²⁶⁾.

When drawing with `TRVReportHelper` ⁽⁸⁰⁴⁾, the default behavior is like in `TRVPrint`, but you can provide a value for the page number in `OnGetPageNumber` ⁽⁸¹¹⁾ 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` ⁽⁹⁵³⁾ (see for detailed information).

Style ⁽³⁰²⁾ of this item type: `rvsPageNumber` (-208)

Methods of `TCustomRichView` ⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- `AddItem` ⁽²⁶⁹⁾ (create `TRVPageNumberItemInfo` ⁽⁹⁵³⁾ object, add it using `AddItem` ⁽²⁶⁹⁾)

The following method returns `TRVPageNumberItemInfo` ⁽⁹⁵³⁾ object

- `GetItem` ⁽²⁹⁶⁾

Methods of `TCustomRichViewEdit` ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem` ⁽⁵²²⁾ (create `TRVPageNumberItemInfo` ⁽⁹⁵³⁾ object, insert it using `InsertItem` ⁽⁵²²⁾)

The following method returns `TRVPageNumberItemInfo` ⁽⁹⁵³⁾ object at the position of caret:

- `GetCurrentItem` ⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx` ⁽⁵⁵⁶⁾;
- `SetItemExtraIntPropertyExEd` ⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrPropertyEx` ⁽⁵⁵⁷⁾;
- `SetItemExtraStrPropertyExEd` ⁽⁵⁶²⁾.

Properties

Fields have all properties of label items ⁽¹⁷⁶⁾.

By default, `Text` ⁽⁹¹⁵⁾ = '{p}'. You can change this value to display another code in `TRichView` and `TRichViewEdit`.

Additionally, it has property:

- `NumberType` ⁽⁹⁵²⁾ – numbering type;

This property is also accessible as `rveipcPageNumberType` ⁽¹⁰⁵¹⁾ property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML ⁽¹²⁷⁾

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` ⁽²⁴⁶⁾.`IgnoreFields` ⁽⁴⁵⁵⁾ = `True`.

4.20 Page counts

When inserted in `TRichView` ⁽²¹⁰⁾ or `TRichViewEdit` ⁽⁴⁶¹⁾, this item is displayed as '{P}'.

When printing with `TRVPrint` ⁽⁷⁵⁹⁾, 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` ⁽⁹⁵³⁾ (see for detailed information).

Style ⁽³⁰²⁾ of this item type: `rvsPageCount` (-209)

Methods of `TCustomRichView` ⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- `AddItem` ⁽²⁶⁹⁾ (create `TRVPageCountItemInfo` ⁽⁹⁵⁴⁾ object, add it using `AddItem` ⁽²⁶⁹⁾)

The following method returns `TRVPageCountItemInfo` ⁽⁹⁵⁴⁾ object

- `GetItem` ⁽²⁹⁶⁾

Methods of `TCustomRichViewEdit` ⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- `InsertItem` ⁽⁵²²⁾ (create `TRVPageCountItemInfo` ⁽⁹⁵⁴⁾ object, insert it using `InsertItem` ⁽⁵²²⁾)

The following method returns `TRVPageCountItemInfo` ⁽⁹⁵⁴⁾ object at the position of caret:

- `GetCurrentItem` ⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- `SetCurrentItemExtraIntPropertyEx` ⁽⁵⁵⁶⁾;
- `SetItemExtraIntPropertyExEd` ⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrPropertyEx` ⁽⁵⁵⁷⁾;

- `SetItemExtraStrPropertyExEd`⁵⁶².

Properties

Fields have all properties of label items¹⁷⁶.

By default, `Text`⁹¹⁵ = '{P}'. You can change this value to display another code in `TRichView` and `TRichViewEdit`.

Additionally, it has property:

- `NumberType`⁹⁵² – numbering type;

This property is also accessible as `rveipcPageNumberType`¹⁰⁵¹ property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML¹²⁷

Fields are saved as a plain text.

RTF and DocX

Fields can be saved to RTF/DocX and loaded from RTF/DocX. You can disallow loading from RTF/DocX by assigning `TRichView.RTFReadProperties`²⁴⁶.`IgnoreFields`⁴⁵⁵ = `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`⁹¹⁷ (see for detailed information).

Style³⁰² 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 versions 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 additional 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 "uses" of the main form unit).

Advantages: the commercial version allows saving and loading equations in HTML and DocX files.

Methods of TCustomRichView⁽²¹⁰⁾

The following viewer-style method adds item of this type to the end of the document:

- AddItem⁽²⁶⁹⁾ (create TRVMathItemInfo⁽⁹¹⁷⁾ object, add it using AddItem⁽²⁶⁹⁾)

The following method returns TRVMathItemInfo⁽⁹¹⁷⁾ object

- GetItem⁽²⁹⁶⁾

Methods of TCustomRichViewEdit⁽⁴⁶¹⁾

The following editor-style method inserts item of this type at the position of caret:

- InsertItem⁽⁵²²⁾ (create TRVMathItemInfo⁽⁹¹⁷⁾ object, insert it using InsertItem⁽⁵²²⁾)

The following method returns TRVMathItemInfo⁽⁹¹⁷⁾ object at the position of caret:

- GetCurrentItem⁽⁵⁰⁸⁾.

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾;
- SetItemExtraIntPropertyExEd⁽⁵⁶²⁾;
- SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾;
- SetItemExtraStrPropertyExEd⁽⁵⁶²⁾.

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⁽¹⁰⁵¹⁾ and rvespcText⁽¹⁰⁵⁶⁾ properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties⁽¹⁰⁰⁰⁾ 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⁽¹⁰⁵¹⁾ and rvespcFontName⁽¹⁰⁵⁶⁾ 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⁽⁹⁷⁴⁾ object.

Saving and Loading

HTML ⁽¹²⁷⁾

- 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(991).SaveMathMLInHTML(969)` 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(991).SaveOMMLInDocX(969)` 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 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 ⁽¹⁹⁰⁾
- Addressing items in table cells (important information) ⁽¹⁹¹⁾
- Table cells ⁽¹⁹²⁾
- Colors and layout of tables ⁽¹⁹³⁾
- Selection in a table ⁽¹⁹⁵⁾
- Cell editing ⁽¹⁹⁷⁾
- Cell merging ⁽¹⁹⁸⁾
- Table operations ⁽¹⁹⁹⁾
- Undo/redo in tables ⁽²⁰²⁾
- Export of tables ⁽²⁰³⁾
- Table resizing ⁽²⁰⁵⁾
- Keys in editor ⁽²⁰⁶⁾

Examples

- Converting table to text ⁽²⁰⁶⁾
- Dividing table ⁽²⁰⁸⁾

Main Classes

- TRVTableItemInfo⁽⁸⁴⁶⁾ – table
- TRVTableCellData⁽⁸⁹⁵⁾ – table cell

5.1 Tables in TRichView

Table is a special kind of RichView items⁽⁸⁵⁾, 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(846);
...
table := TRVTableItemInfo.CreateEx(872)(RowCount, ColumnCount, RVData);
```

where RVData is a richview document where you wish to insert table. Usually this is richview.RVData⁽²⁴⁷⁾ or richviewedit.RVData.

How to append table to RichView⁽²¹⁰⁾

```
table.ParaNo := <Paragraph Style Index>
richview.AddItem(269)(<Table name>, table)
```

where <table name> is any string associated with table, <paragraph style index> is the index in the collection of paragraph styles⁽⁶⁴⁸⁾. 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⁽²⁸⁸⁾ before displaying the document.

How to insert table in RichViewEdit⁽⁴⁶¹⁾

```
richviewedit.InsertItem(522)(<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⁽¹⁷⁴⁾.

The table's paragraph style defines its position: Alignment⁽⁷¹⁹⁾, LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾ (FirstIndent⁽⁷²²⁾ is ignored). The paragraph style can define border⁽⁷²¹⁾ around the table and background⁽⁷²¹⁾. 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⁽¹¹⁴⁾ 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`⁽⁵¹³⁾ and `GetPictureInfo`⁽³¹⁰⁾)

`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`⁽⁴⁸¹⁾.

All methods (and properties) of editor working with current or selected item(s) can work with cell inplace editors

(for example, `ApplyTextStyle`⁽⁴⁹⁴⁾, `SetCurrentTag`⁽⁵⁶⁰⁾, `CurTextStyleNo`⁽⁴⁷⁴⁾, 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`⁽⁴⁸¹⁾).

There are some methods which do not have `ItemNo` parameter:

- `ResizeCurrentControl`⁽⁵⁴³⁾ (alternative to `ResizeControl`⁽⁵⁴²⁾)
- `AdjustControlPlacement2`⁽⁴⁸⁹⁾ (`Control`) (alternative to `AdjustControlPlacement`⁽⁴⁸⁹⁾ (`ItemNo`))
- `SetCurrentItemText`⁽⁵⁵⁷⁾ (`text`); (alternative to `SetItemTextEd`⁽⁵⁶⁴⁾ (`ItemNo`, `text`));
- `GetCurrentItemText`⁽⁵¹¹⁾ (alternative to `GetItemText`⁽³⁰³⁾);

Events with ItemNo Parameter

For events without `RVData`⁽⁹⁵⁷⁾ parameter: when events occur inside cell inplace editor, `ItemNo` is an index of table inside the root editor (for example `OnRVMouseDown`⁽³⁹⁴⁾)

For events having both `RVData` and `ItemNo` parameter: `ItemNo` is an index of item in this `RVData`

Selection

`SelectionExists`⁽³⁴⁹⁾ returns `True`, if something is selected inside cell, or several cells are selected in table. In this case `GetSelectionBound`⁽³¹²⁾ can returns `-1s`.

In editor, you can use `TopLevelEditor.GetSelectionBounds` to get information about selection in the edited cell.

Hypertext ⁹²

Hypertext items can be not only in main document, but in subdocuments (cells ⁸⁹⁵ or cell inplace-editor).

Use `GetJumpPointLocation` ³⁰⁴ 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 ⁸⁷

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` ⁸⁵⁷ [`<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` ⁸⁹⁵. Each RichView control has a similar class inside (`richview.RVData` ²⁴⁷). `Cell` and `richview.RVData` are descendants from the same class ⁹⁵⁷ designed to store formatted RichView document.

Cells also have additional table-related properties (discussed below).

Table Rows

Tables have another property: `Rows` ⁸⁶⁴.

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 ¹⁹¹).

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 ⁽⁹⁰⁸⁾.

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 ⁽¹⁹⁷⁾.

5.4 Colors and layout of tables

Table Colors

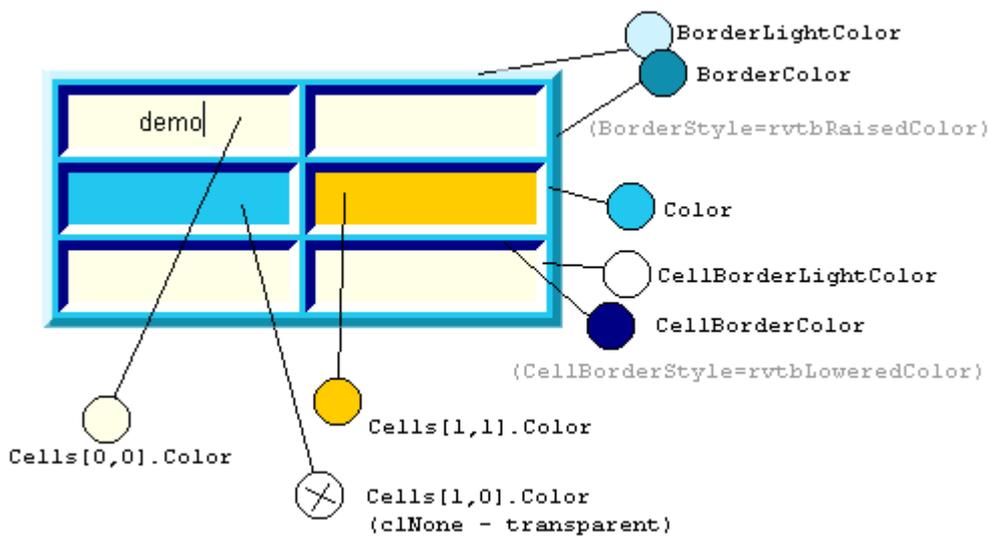


Table Colors

The main color properties of tables (`Color` ⁽⁸⁵⁸⁾, `BorderColor` ⁽⁸⁵¹⁾, `BorderLightColor` ⁽⁸⁵²⁾, `CellBorderColor` ⁽⁸⁵³⁾, `CellBorderLightColor` ⁽⁸⁵⁴⁾) are shown on the figure above.

There are some more color properties related to rules (lines between cells), see below ⁽¹⁹⁴⁾.

Table can be transparent, if its `Color` ⁽⁸⁵⁸⁾ is set to `clNone`, or semitransparent ⁽⁸⁶⁰⁾.

Cells can be individually colored ⁽⁹⁰⁰⁾, with specified opacity ⁽⁹⁰²⁾. You can specify default colors for groups of rows and columns ⁽⁸⁶⁷⁾.

Borders can be hidden, if their widths are set to 0 (see `BorderWidth` ⁽⁸⁵³⁾ and `CellBorderWidth` ⁽⁸⁵⁵⁾). These are default values.

Cell borders can be 3d, like on this figure, or flat (see `BorderStyle` ⁽⁸⁵²⁾ and `CellBorderStyle` ⁽⁸⁵⁴⁾).

You can hide some sides of borders of some cells (see cells' `VisibleBorders`⁽⁹⁰⁶⁾) and set individual color for cell border (`cell.BorderColor`⁽⁸⁹⁹⁾ and `cell.BorderLightColor`⁽⁸⁹⁹⁾).

When a table with invisible border is inserted in `TRichViewEdit`⁽⁴⁶¹⁾, such borders are shown as dotted gray lines (see also table Options⁽⁸⁶¹⁾).

Background Images

Tables and cells can have background images.

See properties of table: `BackgroundImage`⁽⁸⁴⁹⁾, `BackgroundStyle`⁽⁸⁵⁰⁾.

See properties of cell: `BackgroundImage`⁽⁸⁹⁶⁾, `BackgroundStyle`⁽⁸⁹⁷⁾.

Borders and Spacing

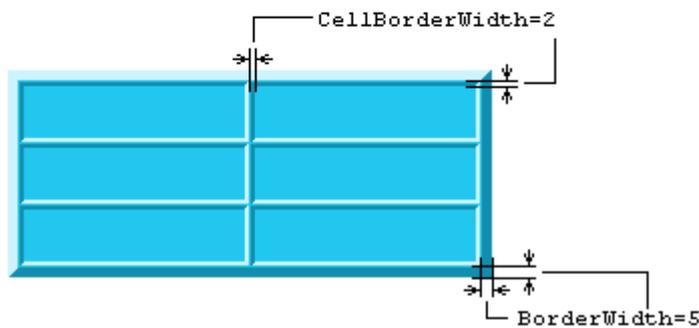
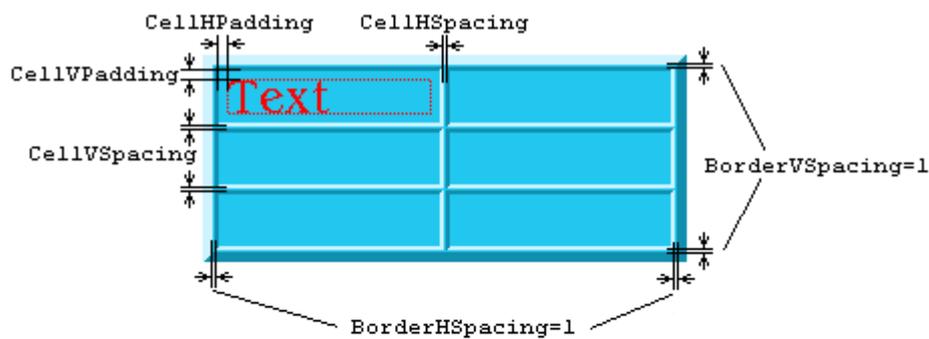


Table Borders

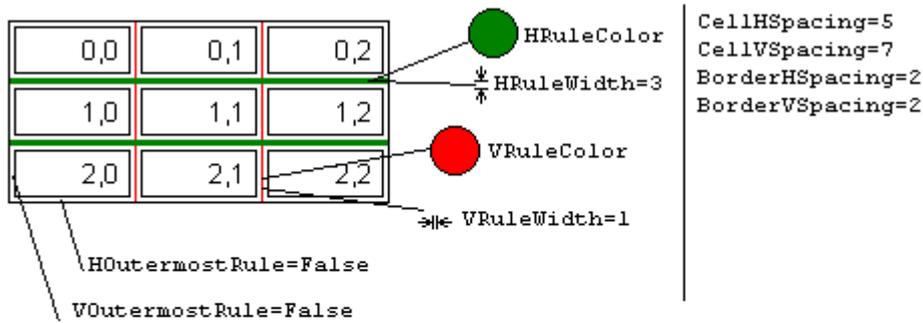
The figure above shows properties affecting the table layout:

- `CellHSpacing`⁽⁸⁵⁶⁾, `CellVSpacing`⁽⁸⁵⁷⁾,
- `BorderHSpacing`⁽⁸⁵¹⁾, `BorderVSpacing`⁽⁸⁵³⁾,
- `BorderWidth`⁽⁸⁵³⁾,
- `CellBorderWidth`⁽⁸⁵⁵⁾,
- `CellHSPadding`⁽⁸⁵⁵⁾, `CellVSPadding`⁽⁸⁵⁷⁾.

(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.



0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

The same table, but
VOutermostRule=True
HOutermostRule=True
BorderHSpacing=5
BorderVSpacing=7

Table Rules

By default rules are invisible.

Rules are defined with properties: `HRuleWidth`⁽⁸⁶⁰⁾, `HRuleColor`⁽⁸⁶⁰⁾, `HOutermostRule`⁽⁸⁵⁹⁾, `VRuleWidth`⁽⁸⁶⁷⁾, `VRuleColor`⁽⁸⁶⁶⁾, `VOutermostRule`⁽⁸⁶⁶⁾.

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`⁽⁸⁵⁵⁾=1, `CellHSpacing`⁽⁸⁵⁶⁾=`CellVSpacing`⁽⁸⁵⁷⁾=0, 1-pixel width borders of adjacent cells will be visually combined in 2-pixel width borders.

Recommended solution:

setting `CellBorderWidth`⁽⁸⁵⁵⁾=1, `CellHSpacing`⁽⁸⁵⁶⁾=`CellVSpacing`⁽⁸⁵⁷⁾=-1.

This example assumes `RVStyle.Units`⁽⁶⁵⁵⁾=`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`⁽⁹⁰⁶⁾ 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`⁽³¹²⁾ (and compare its results with `GetItemNo`⁽³⁰¹⁾ 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`⁽⁸⁷⁸⁾ or `GetNormalizedSelectionBounds`⁽⁸⁷⁷⁾ (the latter is more convenient).

You can select some cells in table using `TRVTableItemInfo.Select`⁽⁸⁸⁴⁾ and remove the selection using `TRVTableItemInfo.Deselect`⁽⁸⁷⁵⁾.

You can select some rows or columns completely: `TRVTableItemInfo.SelectRows`⁽⁸⁸⁴⁾ and `SelectCols`⁽⁸⁸⁴⁾.

You can determine if the given cell is selected or not with the method `IsCellSelected`⁽⁸⁸¹⁾.

Selection Inside Cell

You can get inplace-editor and position for currently edited cell: `GetEditedCell`⁽⁸⁷⁷⁾.

See also: `TRichViewEdit.TopLevelEditor`⁽⁴⁸¹⁾.

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(876)(0,0);
table.Cells(857)[0,0].GetRVData(907).SelectAll(348);
```

Example 2 (does the same)

```
uses CRVData, CRVFData;
var RVData: TCustomRVData(957);
...
RVData := table.Cells[0,0];
RVData := RVData.Edit;
TCustomRVFormattedData(957)(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`⁽⁴⁶¹⁾ methods can be applied to the selected cells (regardless the type of selection):

- `ApplyTextStyle`⁽⁴⁹⁴⁾

- `ApplyParaStyle`⁴⁹⁰
- `ApplyStyleConversion`⁴⁹²
- `ApplyParaStyleConversion`⁴⁹¹
- `DeleteSelection`⁵⁰³.

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 Options⁸⁶¹)

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`⁴⁶¹, not in `TRichView`²¹⁰.

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 contents⁹⁰⁸, use methods of cell's `GetRVData`⁹⁰⁷ instead of methods of the cell itself.

You can initiate editing of cell from code: `table.EditCell`⁸⁷⁶ (see the example in the topic about selection in tables¹⁹⁵) or `Cell.Edit`⁹⁵⁹.

Position of edited cell can be returned by method `GetEditedCell`⁸⁷⁷ and `GetNormalizedSelectionBounds`⁸⁷⁷.

Limitations

This mechanism of cell editing adds some limitations:

You must not destroy inplace editor from inside the following events: `OnKeyDown`, `OnKeyUp`, `OnKeyPress`, `OnRVMouseMove`³⁹⁵, `OnStartDrag`, `OnDragOver`.

Inplace editor can be destroyed by calling `RichViewEdit.Clear`²⁷⁹, or by the methods placing caret/selection outside the editing cell (`RichViewEdit.SetSelectionBounds`³⁶⁵, `table.EditCell`⁸⁷⁶, 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.RichViewEdit1DbClick(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`⁹⁰⁰ and `RowSpan`⁹⁰⁵ properties, similar to html `<td colspan rowspan>`, 1 by default.

If `Cells`⁸⁵⁷[`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=2, RowSpan=3	
ColSpan=1, RowSpan=1		
ColSpan=1, RowSpan=1		

ColSpan and RowSpan example

Example: iterating through all cells of table:

```
for r := 0 to table.RowCount864-1 do
  for c := 0 to table.ColCount858-1 do
    if table.Cells857[r,c] <> nil then
      table.Cells857[r,c].Color900 := 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`⁸⁶⁴.`GetMainCell` method:

```
function GetMainCell912(ARow, ACol: Integer;
  out MRow, MCol: Integer): TRVTableCellData895;
```

Methods for merging cells:

- `MergeCells`⁸⁸¹,

- MergeSelectedCells⁽⁸⁸¹⁾

Inverse operation:

- UnmergeCells⁽⁸⁹⁰⁾,
- UnmergeSelectedCells⁽⁸⁹¹⁾

Splitting cells:

- SplitSelectedCellsHorizontally⁽⁸⁹⁰⁾,
- SplitSelectedCellsVertically⁽⁸⁹⁰⁾

5.8 Table operations

Operations on Selection⁽¹⁹⁵⁾

Inserting rows and columns:

- InsertColsLeft⁽⁸⁷⁹⁾,
- InsertColsRight⁽⁸⁷⁹⁾,
- InsertRowsAbove⁽⁸⁸⁰⁾,
- InsertRowsBelow⁽⁸⁸⁰⁾.

Deleting rows and columns:

- DeleteSelectedCols⁽⁸⁷⁴⁾,
- DeleteSelectedRows⁽⁸⁷⁵⁾,

Cell merging and unmerging⁽¹⁹⁸⁾:

- MergeSelectedCells⁽⁸⁸²⁾ (see also CanMergeSelectedCells⁽⁸⁷¹⁾),
- UnmergeSelectedCells⁽⁸⁹¹⁾,
- SplitSelectedCellsHorizontally⁽⁸⁹⁰⁾,
- SplitSelectedCellsVertically⁽⁸⁹⁰⁾

Other Operations

Inserting rows and columns:

- InsertCols⁽⁸⁷⁸⁾,
- InsertRows⁽⁸⁸⁰⁾.

Deleting rows and columns:

- DeleteCols⁽⁸⁷³⁾,
- DeleteRows⁽⁸⁷⁴⁾.

Cell merging and unmerging⁽¹⁹⁸⁾:

- MergeCells⁽⁸⁸¹⁾ (also CanMergeCells⁽⁸⁷⁰⁾)
- UnmergeCells⁽⁸⁹⁰⁾,
- DeleteEmptyRows⁽⁸⁷³⁾, DeleteEmptyCols⁽⁸⁷³⁾ (use after cell merging).

Moving rows

- MoveRows⁽⁸⁸²⁾.

Assigning Table Properties (As Editing Operations)

Methods for assigning table properties:

- SetTableVisibleBorders⁽⁸⁸⁹⁾.

Methods for assigning row properties:

- SetRowVAlign, SetRowKeepTogether, SetRowPageBreakBefore⁽⁸⁸⁹⁾.

Methods for assigning cell properties:

- SetCellColor⁽⁸⁸⁷⁾, SetCellBorderColor⁽⁸⁸⁶⁾, SetCellBorderLightColor⁽⁸⁸⁶⁾;
- SetCellVAlign⁽⁸⁸⁸⁾;
- SetCellRotation⁽⁸⁸⁸⁾;
- SetCellTag⁽⁸⁸⁸⁾;
- SetCellBestWidth⁽⁸⁸⁶⁾, SetCellBestHeight⁽⁸⁸⁶⁾;
- SetCellBackgroundImage⁽⁸⁸⁵⁾, SetCellBackgroundImageFileName⁽⁸⁸⁵⁾, SetCellBackgroundStyle⁽⁸⁸⁵⁾;
- SetCellHint⁽⁸⁸⁷⁾;
- SetCellVisibleBorders⁽⁸⁸⁹⁾;
- SetCellOpacity⁽⁸⁸⁷⁾;
- SetCellOptions⁽⁸⁸⁷⁾.

Operations in Viewer and Editor

If the operations above are performed before inserting the table, no additional actions are required.

If these operations are performed on table inserted in TRichView⁽²¹⁰⁾, you need to call Format⁽²⁸⁸⁾ method after to update the document view.

If these operations are performed on table inserted in TRichViewEdit⁽⁴⁶¹⁾, a special sequence of steps is required:

1. checking RichViewEdit.CanChange⁽⁴⁹⁸⁾ (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⁽⁴⁹⁵⁾ for table
4. performing operations
5. calling rve.EndItemModify⁽⁵⁰³⁾
6. calling rve.Change⁽⁴⁹⁹⁾

The example is below.

You can group several actions so that they will be undone/redone as whole, using SetUndoGroupMode⁽⁵⁶⁷⁾

In order to perform operation(s) on table, you need to get table object. You can do it using method

```
function TRichView.GetItem(296)(ItemNo: Integer): TCustomRVItemInfo;
```

TCustomRVItemInfo⁽⁸⁴⁴⁾ is an ancestor class for all items of RichView, including table (TRVTableItemInfo⁽⁸⁴⁶⁾). 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⁽³⁰²⁾(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(508): 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(509) (  
    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(844);  
    table: TRVTableItemInfo(846);  
    Data: Integer;  
    rve: TCustomRichViewEdit(461);  
    ItemNo: Integer;  
begin  
    if not MyRichViewEdit.CanChange(498) or  
        not RichViewEdit1.GetCurrentItemEx(509)(TRVTableItemInfo, rve,  
            item) then  
        exit;  
    table := TRVTableItemInfo(item);  
    ItemNo := rve.GetItemNo(301)(table);  
    rve.BeginItemModify(495)(ItemNo, Data);  
    // performing some operation, for example  
    // table.InsertRowsBelow(1);  
    // or table.CellPadding := 10  
    rve.EndItemModify(503)(ItemNo, Data);  
    rve.Change(499);  
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⁽⁴⁶¹⁾.

Operations will not be undone if:

- table was created, but not inserted yet;
- table was inserted in RichView⁽²¹⁰⁾;
- table was inserted in RichViewEdit with UndoLimit⁽⁴⁸¹⁾=0.

Example 1

Two operations (inserting table and changing its color) will be undone in two steps

```
table := TRVTableItemInfo.CreateEx(872)(4,3, RichViewEdit1.RVData(247));
if RichViewEdit1.InsertItem(522)('', table) then
  table.Color(900) := 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⁽²⁷⁰⁾, for example)

These methods must not be mixed with editor-style methods (such as InsertText⁽⁵³²⁾) 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⁽²⁷⁹⁾ 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⁽⁴⁸¹⁾ of editor = 0 (undo mechanism turned off). But Format⁽²⁸⁸⁾ must be called before calling any editor-style method.
- After ClearUndo⁽⁵⁰¹⁾ method. But Format⁽²⁸⁸⁾ must be called before calling any editor-style method.

Undo of Assignment to Properties

Assigning values to cell properties (and to VAlign⁽⁹¹⁰⁾ property of rows) will not be undone.

But there are some special methods.

```
table.Cells(857)[r,c].Color(900) := clRed; // will not be undone
table.SetCellColor(887)(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`⁽⁴⁹⁵⁾ / `EndItemModify`⁽⁵⁰³⁾.

For example

```
var item: TCustomRVItemInfo;
    table: TRVTableItemInfo;
    Data: Integer;
    rve: TCustomRichViewEdit;
    ItemNo: Integer;
begin
    if not RichViewEdit1.CanChange(498) or
        not RichViewEdit1.GetCurrentItemEx(509)(TRVTableItemInfo, rve,
            item) then
        exit;
    table := TRVTableItemInfo(item);
    ItemNo := rve.GetItemNo(301)(table);
    rve.BeginItemModify(495)(ItemNo, Data);
    table.SetRowVAlign(rcvBottom, 1);
    table.SetCellColor(clRed, 1, 1);
    rve.EndItemModify(503)(ItemNo, Data);
    rve.Change(499);
end;
```

Assignments to all table properties can be undone (but still the sequence `BeginItemModify`-`EndItemModify`-`Change` is required). The only exception is `VisibleBorders`⁽⁸⁶⁵⁾ property, use `SetTableVisibleBorders`⁽⁸⁸⁹⁾ 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`⁽⁸⁸⁹⁾.

Methods for assigning row properties:

- `SetRowVAlign`⁽⁸⁸⁹⁾.

Methods for assigning cell properties:

- `SetCellColor`⁽⁸⁸⁷⁾, `SetCellBorderColor`⁽⁸⁸⁶⁾, `SetCellBorderLightColor`⁽⁸⁸⁶⁾;
- `SetCellVAlign`⁽⁸⁸⁸⁾;
- `SetCellBestWidth`⁽⁸⁸⁶⁾, `SetCellBestHeight`⁽⁸⁸⁶⁾;
- `SetCellBackgroundImage`⁽⁸⁸⁵⁾, `SetCellBackgroundImageFileName`⁽⁸⁸⁵⁾, `SetCellBackgroundStyle`⁽⁸⁸⁵⁾;
- `SetCellHint`⁽⁸⁸⁷⁾;
- `SetCellVisibleBorders`⁽⁸⁸⁹⁾.

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`⁽⁸⁶⁵⁾ – string to write after each row;
- `TextColSeparator`⁽⁸⁶⁵⁾ – 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`⁽⁸⁵⁷⁾[*r,c*].`BestWidth`⁽⁸⁵⁰⁾ (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`⁽⁸⁵⁰⁾ = 0 and `BestWidth`⁽⁸⁵⁰⁾ > 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`⁽⁸⁵⁰⁾ by including `rvtoRTFSaveCellPixelBestWidth` to `table.Options`⁽⁸⁶¹⁾; however, this is not recommended for fixed tables.

If you include `rvtoRTFAllowAutofit` in `table.Options`⁽⁸⁶¹⁾, this table will be saved as fit-by-content table (but only if `table.BestWidth`⁽⁸⁵⁰⁾ = 0). For such tables, you can safely include `rvtoRTFSaveCellPixelBestWidth` to `table.Options`⁽⁸⁶¹⁾.

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`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = `rvhtmlstNormal`, or `rvcsssoForceNonTextCSS` in `TRichView.HTMLSaveProperties`⁽²³⁷⁾.`CSSOptions`⁽⁴³⁰⁾). 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 `rvmdcTables` in `TRichView.MarkdownProperties`⁽²³⁹⁾.`Extensions`⁽⁴⁴⁴⁾ (included by default).

Otherwise, content of table cells is saved as normal paragraphs.

See Also...

TRichView methods for saving and loading ⁽¹²²⁾

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` ⁽⁸⁶¹⁾)
- `table.BestWidth` ⁽⁸⁵⁰⁾
- cells `BestWidth` ⁽⁸⁹⁸⁾

Heights are calculated basing on:

- heights of cells content (if *rvtoIgnoreContentHeight* is not in `table.Options` ⁽⁸⁶¹⁾ and `cell.IgnoreContentHeight` ⁽⁹⁰¹⁾ = *False*)
- cells `BestHeight` ⁽⁸⁹⁸⁾

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` ⁽⁸⁶¹⁾);
2. if `table.BestWidth` ⁽⁸⁵⁰⁾ <> 0, it defines width of the table (but table can be wider because of the rule 1)
3. if `table.BestWidth` ⁽⁸⁵⁰⁾ = 0 then
 - if all columns have cells with `BestWidth` ⁽⁸⁹⁸⁾s in `TRVStyleUnits` ⁽¹⁰²⁷⁾, i.e. pixels or twips or EMU (and do not have cells with `BestWidth` ⁽⁸⁹⁸⁾s in %), width of table is calculated basing on cells `BestWidth` ⁽⁸⁹⁸⁾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` ⁽⁸⁵⁰⁾ = -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` ⁽⁸⁶¹⁾, users will not be able to resize it.

When user resizes table with mouse, he/she actually changes values of `BestWidth` ⁽⁸⁹⁸⁾ or `BestHeight` ⁽⁸⁹⁸⁾ properties of cells.

When resizing columns with mouse, and columns to the left and to the right do not have cells with `BestWidth` ⁸⁹⁸ s in %, `RichViewEdit` adjusts `BestWidth` ⁸⁹⁸ 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` ⁸⁹⁸ s of the column to the left.

5.12 Keys in editor

Keyboard processing

When inserted in `TRichViewEdit` ⁴⁶¹, 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` ¹⁰⁴⁹).

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` ¹⁰⁴⁹).

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 ¹¹⁵ by the user.

```

procedure ConvertTableToText(RichViewEdit: TCustomRichViewEdit 461);
var table: TRVTableItemInfo 846;
    rv: TRichView 210;
    rve: TCustomRichViewEdit 461;
    Stream: TMemoryStream;
    ItemNo,r,c: Integer;
begin
    if not RichViewEdit.GetCurrentItemEx 509 (TRVTableItemInfo 846, rve,
        TCustomRVItemInfo 844 (table)) then
        exit;
    rv := TRichView 210.Create 283 (nil);
    try
        rv.Visible := False;
        rv.Parent := RichViewEdit.Parent;
        rv.Style 253 := RichViewEdit.Style 253;
        rv.RVFTextStylesReadMode 251 := rvf_sIgnore;
        rv.RVFParaStylesReadMode 251 := rvf_sIgnore;
        for r := 0 to table.RowCount 864 -1 do
            for c := 0 to table.ColCount 858 -1 do
                if table.Cells 857 [r,c] <> nil then
                    begin
                        Stream := TMemoryStream.Create;

```

```

    try
        table.Cells857[r,c].GetRVData907.SaveRVFToStream344(Stream,
            False, clNone, nil, nil);
        Stream.Position := 0;
        rv.InsertRVFFromStream316(Stream, rv.ItemCount237)
    finally
        Stream.Free;
    end;
end;
end;
// GetMyItemNo is like rve.GetItemNo301(table), but faster
ItemNo := table.GetMyItemNo;
Stream := TMemoryStream.Create;
try
    rv.SaveRVFToStream344(Stream, False);
    Stream.Position := 0;
    rve.SetSelectionBounds365(ItemNo, 0, ItemNo, 1);
    rve.InsertRVFFromStreamEd528(Stream);
finally
    Stream.Free;
end;
finally
    rv.Free;
end;
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->InsertRVFFFromStreamEd(Stream);
delete Stream;
delete rv;
}

```

5.14 Example: dividing table

```

function SplitTable(rve: TCustomRichViewEdit461;
  table: TRVTableItemInfo846; RowCountInFirstTable: Integer;
  PageBreak: Boolean): Boolean;
var r, c, mr, mc: Integer;
    newtable: TRVTableItemInfo846;
    Stream: TMemoryStream;
begin
  Result := False;
  if RowCountInFirstTable >= table.RowCount864 then
    exit;
  for c := 0 to table.ColCount858 - 1 do
    if table.Cells857[RowCountInFirstTable, c] = nil then
      begin
        table.Rows864.GetMainCell912(RowCountInFirstTable, c, mr, mc);
        if mr < RowCountInFirstTable then
          exit;
        end;
      Stream := TMemoryStream.Create;
      table.SaveRowsToStream883(Stream, RowCountInFirstTable,
        table.RowCount864 - RowCountInFirstTable);
      Stream.Position := 0;
      newtable := TRVTableItemInfo846.CreateEx(0, 0, rve.RVData);
      newtable.LoadFromStream881(Stream);
      if PageBreak then
        newtable.PageBreakBefore := True;
      Stream.Free;
      rve.BeginUndoGroup497(rvutModifyItem);
      rve.SetUndoGroupMode567(True);
      try
        table.DeleteRows874(RowCountInFirstTable,
          table.RowCount864 - RowCountInFirstTable, True);
        // GetMyItemNo is the same as rve.GetItemNo301(table), but faster
        rve.SetSelectionBounds365(table.GetMyItemNo, 1, table.GetMyItemNo, 1);
        rve.InsertItem522(' ', newtable);
      finally
        rve.SetUndoGroupMode567(False);
      end;
      Result := True;
    end;
end;

```

Parameters:

rve – editor directly containing **table**;

table – the table to divide.

These two parameters are usually a result of `RichViewEdit1.GetCurrentItemEx`⁽⁵⁰⁹⁾.

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`⁽⁸⁶⁴⁾ - **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⁽¹¹⁵⁾ by the user.

6 Components

Main Components

Icon	Class Name	Description
	TRVStyle ⁽⁶³⁰⁾	Component for defining visual appearance of TRichView ⁽²¹⁰⁾ , TRichViewEdit ⁽⁶⁰⁶⁾ , TDBRichView ⁽⁵⁹⁴⁾ , TDBRichViewEdit ⁽⁶⁰⁶⁾ controls. It contains some properties which can be used by several TRichView controls.
	TRichView ⁽²¹⁰⁾	Component for displaying rich text documents
	TRichViewEdit ⁽⁴⁶¹⁾	Component for editing rich text documents
	TRVPrint ⁽⁷⁵⁹⁾	Component for printing documents contained in TRichView ⁽²¹⁰⁾ , TRichViewEdit ⁽⁶⁰⁶⁾ , TDBRichView ⁽⁵⁹⁴⁾ , TDBRichViewEdit ⁽⁶⁰⁶⁾
	TRVPrintPreview ⁽⁶²⁵⁾	Component allowing to show document as it will be printed by TRVPrint
	TRVReportHelper ⁽⁸⁰⁴⁾	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter ⁽⁷⁹⁵⁾	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL; obsolete]
	TRVSpellChecker ⁽⁷⁸⁴⁾	Component to check spelling in TRichView ⁽²¹⁰⁾ , TRichViewEdit ⁽⁶⁰⁶⁾ , TDBRichView ⁽⁵⁹⁴⁾ , TDBRichViewEdit ⁽⁶⁰⁶⁾ , TSBRichViewEdit, TDBSRichViewEdit controls

Data-Aware Components [VCL and LCL]

Icon	Class Name	Description
------	------------	-------------

	TDBRichView ⁽⁵⁹⁴⁾	Data-aware version of TRichView [VCL and LCL]
	TDBRichViewEdit ⁽⁶⁰⁶⁾	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVDataSourceLink ⁽⁸¹¹⁾	Component allowing assigning TDataSource component to controls inserted in TRichView ⁽²¹⁰⁾ , TRichViewEdit ⁽⁶⁰⁶⁾ , TDBRichView ⁽⁵⁹⁴⁾ , TDBRichViewEdit ⁽⁶⁰⁶⁾ , TSRichViewEdit, TDBSRichViewEdit controls [VCL and LCL]

6.1 Visual Components

TRichView Components⁽²⁰⁹⁾ | Visual Components

Icon	Class Name	Description
	TRichView ⁽²¹⁰⁾	Component for displaying rich text documents
	TRichViewEdit ⁽⁴⁶¹⁾	Component for editing rich text documents
	TDBRichView ⁽⁵⁹⁴⁾	Data-aware version of TRichView
	TDBRichViewEdit ⁽⁶⁰⁶⁾	Data-aware version of TRichViewEdit [VCL and LCL]
	TRVPrintPreview ⁽⁶²⁵⁾	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(813))
TRichView = class(TCustomRichView)
```

TRichView does not display its content if it is not linked to TRVStyle component via Style⁽²⁵³⁾ property.

The most important property settings can be done in the component editor⁽²¹⁸⁾ 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⁽⁸¹³⁾
TCustomRichView

► Hierarchy (FireMonkey)

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBar
TCustomRVControl
TRVScroller⁽⁸¹³⁾
TCustomRichView

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

<ul style="list-style-type: none"> ❖ General⁽²¹¹⁾ ❖ Background⁽²¹¹⁾ ❖ Appending Items⁽²¹²⁾ ❖ Hypertext⁽²¹²⁾ ❖ Style templates⁽²¹²⁾ ❖ Scrolling, margins, sizes⁽²¹³⁾ ❖ Selection and Clipboard⁽²¹³⁾ ❖ Saving and loading as RVF⁽²¹⁴⁾ ❖ Other methods for saving and loading⁽²¹⁴⁾ ❖ LiveBindings and DB⁽²¹⁵⁾ 	<ul style="list-style-type: none"> ❖ Information about items⁽²¹⁵⁾ ❖ Modifying items⁽²¹⁶⁾ ❖ Checkpoints⁽²¹⁶⁾ ❖ Mouse events⁽²¹⁷⁾ ❖ Live spelling check⁽²¹⁷⁾ ❖ Picture animation⁽²¹⁷⁾ ❖ Working in palette mode⁽²¹⁷⁾ ❖ Custom Drawing⁽²¹⁷⁾ ❖ Other properties, methods and events⁽²¹⁷⁾
--	--

General properties

- Style⁽²⁵³⁾ – link to RVStyle component defining a lot of properties which can be used in several RichViews;
- Options⁽²⁴⁰⁾ – some important options;
- ItemCount⁽²³⁷⁾ – number of items in the document.

Properties defining background

- BackgroundBitmap⁽²²²⁾;
- BackgroundStyle⁽²²³⁾;
- Color⁽²²⁶⁾.

Appending items (Add*** methods)

Main methods for appending items:

- AddNL⁽²⁷⁰⁾ adds one text item;
- AddBreak⁽²⁶²⁾ adds break (horizontal line);
- AddBullet⁽²⁶³⁾ adds bullet (image from ImageList);
- AddHotspot⁽²⁶⁸⁾ adds hotspot (image from ImageList - hypertext link);
- AddPicture⁽²⁷²⁾ adds picture;
- AddHotPicture⁽²⁶⁷⁾ adds picture - hypertext link;
- AddControl⁽²⁶⁵⁾ adds Delphi/C++Builder control;
- AddTab⁽²⁷³⁾ adds tabulator;
- AddItem⁽²⁶⁹⁾ – general method for appending items (usually used for tables⁽¹⁹⁰⁾);
- also AddCheckpoint⁽²⁶⁴⁾ – adds checkpoint which will be associated with next added item.

Additional methods for appending items:

- Add⁽²⁷⁰⁾ adds one text item in last paragraph;
- AddFmt⁽²⁶⁶⁾ adds one formatted string assembled from a Pascal format string and an array arguments;
- AddTextNL, -A -W⁽²⁷⁴⁾ add several text items from text, separated by CR+LF characters;

Also

- AppendFrom⁽²⁷⁵⁾ copies content from another RichView;
- InsertRVFFromStream⁽³¹⁶⁾
- AppendRVFFromStream⁽²⁷⁶⁾.

See also

- Clear⁽²⁷⁹⁾ clears document;
- SetAddParagraphMode⁽³⁵⁰⁾;
- Format⁽²⁸⁸⁾, FormatTail⁽²⁸⁸⁾ format the document;
- Reformat⁽³³²⁾ reformats the document.

Hypertext

- event OnJump⁽³⁸²⁾ occurs when user clicks hyperlink;
- event OnRVMouseMove⁽³⁹⁵⁾ occurs when mouse pointer moves over hyperlink;
- events OnWriteHyperlink⁽⁴¹¹⁾ and OnReadHyperlink⁽³⁸⁸⁾ occur when document is saved or loaded as RTF, DocX or HTML.
- property FirstJumpNo⁽²³⁶⁾ – use to set hypertext id for the first hypertext item;
- GetJumpPointLocation⁽³⁰⁴⁾ should be used instead of GetJumpPointItemNo in applications working with tables⁽¹⁹⁰⁾;
- GetJumpPointY⁽³⁰⁵⁾ returns Y coordinate of item with the specified hypertext id.

Style templates

Properties:

- UseStyleTemplates⁽²⁵⁶⁾ allows or disallows using Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾.
- StyleTemplateInsertMode⁽²⁵⁴⁾ specifies how style templates are used when inserting RTF or RVF content.

Events:

- OnStyleTemplatesChange⁽⁴¹⁰⁾ occurs then the collection of style templates has been modified.

Scrolling, margins and sizes

Properties:

- `LeftMargin`⁽²³⁸⁾, `RightMargin`⁽²⁴⁴⁾, `TopMargin`⁽²⁵⁵⁾, `BottomMargin`⁽²²⁵⁾ – margins;
- `MaxTextWidth`⁽²³⁹⁾ – maximal width for word wrapping;
- `MinTextWidth`⁽²⁴⁰⁾ – minimal width for word wrapping (minimal width of scrollable area, not including margins);
- `DocumentHeight`⁽²³³⁾ – height of document (height of scrollable area);
- `VAlign`⁽²⁵⁷⁾ – vertical alignment of document;
- `VScrollVisible`⁽⁸¹⁹⁾ – hides/shows vertical scrollbar;
- `HScrollVisible`⁽⁸¹⁶⁾ – hides/shows horizontal scrollbar;
- `VSmallStep`⁽²⁵⁷⁾ – number of pixels in one step of vertical scrollbar;
- `FirstItemVisible`⁽²³⁶⁾ – the first visible item;
- `LastItemVisible`⁽²³⁸⁾ – the last visible item;
- `MaxLength`⁽²³⁹⁾ – maximal count of characters per line;
- `WordWrap`⁽²⁵⁸⁾ allows to disable word wrapping;
- `Tracking`⁽⁸¹⁸⁾ – whether RichView repaints while its scrollbar thumb tab is dragged with the mouse.

Methods:

- `ClientToDocument`⁽²⁸⁰⁾ converts client coordinates to document coordinates

Events:

- `OnVScrolled`⁽⁸²¹⁾ occurs when document is scrolled vertically
- `OnHScrolled`⁽⁸²¹⁾ occurs when document is scrolled horizontally

Also:

- property `CPEventKind`⁽²²⁷⁾
- event `OnCheckPointVisible`⁽³⁷²⁾
- method `GetItemCoords`⁽²⁹⁸⁾

Methods for working with selection and the Clipboard

- `SelectionExists`⁽³⁴⁹⁾ returns "is something selected?"
- `Copy`⁽²⁸¹⁾ copies selection to the Clipboard in all available formats
- `CopyDef`⁽²⁸¹⁾ copies selection to the Clipboard using formats specified in `Options`⁽²⁴⁰⁾;
- `CopyImage` copies selected image to the Clipboard;
- `CopyTextA`⁽²⁸³⁾ copies selection as text to the Clipboard;
- `CopyTextW`⁽²⁸³⁾ copies selection to the Clipboard as Unicode text;
- `CopyRVF`⁽²⁸²⁾ copies selection as RVF to the Clipboard;
- `CopyRTF`⁽²⁸²⁾ copies selection as RTF (Rich Text Format);
- `GetSelectedImage`⁽³¹²⁾ returns selected image
- `GetSelText`⁽³¹³⁾ returns selection as text
- `GetSelTextW`⁽³¹³⁾ returns selection as Unicode string;
- `GetOffsBeforeItem`⁽³⁰⁹⁾, `GetOffsAfterItem`⁽³⁰⁸⁾ return offset before/after the given item; useful with `SetSelectionBounds`⁽³⁶⁵⁾;
- `SelectAll`⁽³⁴⁸⁾ selects the whole document for copying to the Clipboard;

- `SetSelectionBounds`⁽³⁶⁵⁾ selects the specified items;
- `SelectWordAt`⁽³⁴⁹⁾ selects the word at (X,Y);
- `SelectControl`⁽³⁴⁸⁾ selects a visual control;
- `Deselect`⁽²⁸⁶⁾ clears selection (not contents, but bounds);
- `GetSelectionBounds`⁽³¹²⁾ returns bounds of selection;
- event `OnSelect`⁽⁴⁰⁸⁾ occurs when selection bounds are changed;
- event `OnCopy`⁽³⁷⁴⁾ allows to copy selection to the Clipboard in custom formats;
- property `SelectionHandlesVisible`⁽²⁵³⁾ shows or hides selection handles for a touch screen.

Properties, methods and events for loading and saving RVF

Properties:

- `RVFOptions`⁽²⁴⁷⁾ – options for saving and loading RVF;
- `RVFWarnings`⁽²⁵¹⁾ describes warnings or errors occurred during the last reading of RVF.

Events:

- `OnRVFControlNeeded`⁽³⁹²⁾ requests inserted control when reading RVF, if RVF was saved without controls' "bodies";
- `OnRVFImageListNeeded`⁽³⁹³⁾ requests `ImageList` when reading bullet⁽¹⁷⁰⁾ or hotspot⁽¹⁷²⁾ from RVF;
- `OnRVFPictureNeeded`⁽³⁹³⁾ requests image when reading RVF, if RVF was saved without images' "bodies".

Methods:

- `SaveRVF`⁽³⁴⁴⁾ saves document or selection as RVF to the file;
- `SaveRVFToStream`⁽³⁴⁴⁾ saves document or selection as RVF to the stream;
- `LoadRVF`⁽³²⁸⁾ loads RVF from the file;
- `LoadRVFFromStream`⁽³²⁹⁾ loads RVF from the stream;
- `CopyRVF`⁽²⁸²⁾ copies selection as RVF to the Clipboard;
- `InsertRVFFromStream`⁽³¹⁶⁾ inserts RVF from the stream;
- `AppendRVFFromStream`⁽²⁷⁶⁾ appends RVF from the stream using the specified paragraph style for the first RVF item.

Also:

- methods `Copy`⁽²⁸¹⁾, `CopyDef`⁽²⁸¹⁾.

Other saving and loading

Methods:

- `SetHeader`⁽³⁵⁶⁾, `SetFooter`⁽³⁵⁵⁾ 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`⁽³²¹⁾ load document in RVF, RTF, HTML, DocX, Markdown or text format (format detection) from a stream;
- `LoadFromFile`, `LoadFromFileEx`⁽³²⁰⁾ load document in RVF, RTF, HTML, DocX, Markdown or text format (format detection) from a file;
- `LoadHTML`⁽³²²⁾, `LoadHTMLFromStream`⁽³²⁴⁾ appends a file or a stream in HTML format;
- `LoadText`⁽³³⁰⁾ appends text from a file;
- `LoadTextW`⁽³³⁰⁾ appends text from a file in Unicode (UTF-16) encoding;
- `LoadRTF`⁽³²⁶⁾, `LoadRTFFromStream`⁽³²⁷⁾ appends a file or a stream in RTF (Rich Text Format);

- LoadMarkdown⁽³²⁵⁾, LoadMarkdownFromStream⁽³²⁵⁾ appends a file or a stream in **Markdown** format;
- SaveHTML⁽³³⁵⁾, SaveHTMLToStream⁽³³⁸⁾ save the document in an HTML file/stream;
- SaveText⁽³⁴⁵⁾, SaveTextToStream⁽³⁴⁵⁾ save the document in a file/stream as ANSI text;
- SaveTextW⁽³⁴⁵⁾, SaveTextToStreamW⁽³⁴⁵⁾ save the document in a file/stream as a Unicode text;
- SaveDocX⁽³³³⁾, SaveDocXToStream⁽³³⁴⁾ save the document (or selection) in a file/stream as DocX (Microsoft Word format);
- SaveRTF⁽³⁴³⁾, SaveRTFToStream⁽³⁴⁴⁾ save the document (or selection) in a file/stream as RTF.
- SaveMarkdown⁽³⁴⁰⁾, SaveMarkdownToStream⁽³⁴¹⁾ save the document (or selection) in a file/stream as **Markdown**.

Properties:

- RTFOptions⁽²⁴⁵⁾ – options for DocX and RTF export.
- RTFReadProperties⁽²⁴⁶⁾ – properties for DocX and RTF import.
- HTMLReadProperties⁽²³⁷⁾ – properties for HTML import.
- HTMLSaveProperties⁽²³⁷⁾ – properties for HTML export.
- MarkdownProperties⁽²³⁹⁾ – properties for **Markdown** import and export.

Events:

- OnSaveComponentToFile⁽³⁹⁷⁾ allows to save inserted controls in DocX, RTF, Markdown, HTML, and text files;
- OnWriteHyperlink⁽⁴¹¹⁾, OnReadHyperlink⁽³⁸⁸⁾ allow to customize saving/loading of hypertext links in HTML, Markdown, RTF and DocX files;
- OnWriteObjectProperties⁽⁴¹³⁾ allows saving additional properties of non-text objects in HTML, RTF and DocX files;
- OnReadMergeField⁽³⁹⁰⁾ allow importing merge fields from RTF;
- OnHTMLSaveImage⁽³⁷⁵⁾, OnSaveImage2⁽⁴⁰²⁾ allow to save images with HTML in custom formats and with custom file names;
- OnSaveItemToFile⁽⁴⁰⁴⁾ allows to save any item in RTF, DocX, Markdown, HTML or text files in your own format;
- OnImportPicture⁽³⁷⁸⁾ allows custom processing of external images when importing RTF, DocX, Markdown, HTML;
- OnImportFile⁽³⁷⁸⁾ allows custom processing of external files when importing HTML;
- OnAssignImageFileName⁽³⁷¹⁾ allows changing file names assigned to inserted images when importing RTF, DocX or Markdown.

LiveBindings and Databases

Properties:

- Document⁽²³²⁾ allows linking this TRichView control to TBlobField field of DB (in Delphi XE2+)

Events:

- OnNewDocument⁽³⁸⁴⁾ occurs when creating or before loading a document from a DB field
- OnLoadDocument⁽³⁸⁴⁾ occurs after loading a document from a DB field
- OnLoadCustomFormat⁽³⁸³⁾ allows loading documents from a DB field in custom formats

Information about items (Get***Info methods)

- GetItemStyle⁽³⁰²⁾ returns a type of item;
- GetTextInfo⁽³¹³⁾ – information about text item;

- GetBreakInfo⁽²⁸⁹⁾ – about "break";
- GetBulletInfo⁽²⁸⁹⁾ – about "bullet";
- GetHotspotInfo⁽²⁹⁵⁾ – about "hotspot";
- GetPictureInfo⁽³¹⁰⁾ – about picture or "hot-picture";
- GetControllInfo⁽²⁹³⁾ – about inserted control;
- GetItemExtraIntProperty[Ex]⁽³⁰⁰⁾,
GetItemExtraStrProperty[Ex]⁽³⁰⁰⁾ return value of extra item property;
- GetItem⁽²⁹⁶⁾ – general method for obtaining items (usually used for tables⁽¹⁹⁰⁾);

- GetItemCheckpoint⁽²⁹⁸⁾ – about item's "checkpoint";
- GetItemVAlign⁽³⁰³⁾ – about item's position relative to line;
- GetItemTag⁽³⁰²⁾ – about item's "tag";
- GetItemText⁽³⁰³⁾ – about text/name of item;
- IsParaStart⁽³¹⁸⁾ returns "is this item a first item in paragraph?";
- IsFromNewLine⁽³¹⁷⁾ returns "is this item a first item in paragraph?";
- GetItemPara⁽³⁰¹⁾ returns an index of paragraph containing a specified item;
- GetItemNo⁽³⁰¹⁾ returns item index for object representing item (usually used for tables⁽¹⁹⁰⁾);
- GetItemCoords⁽²⁹⁸⁾ returns coordinates of item relative to document area, GetItemClientCoords⁽²⁹⁸⁾ returns coordinates of item relative to client area of RichView window;
- properties PageBreaksBeforeItems⁽²⁴⁴⁾, ClearLeft⁽²²⁵⁾, ClearRight⁽²²⁶⁾.

Modifying items (Set***Info methods)

- SetBreakInfo⁽³⁵¹⁾ changes *break*⁽¹⁶⁷⁾;
- SetBulletInfo⁽³⁵²⁾ changes *bullet*⁽¹⁷⁰⁾;
- SetHotspotInfo⁽³⁵⁷⁾ changes *hotspot*⁽¹⁷²⁾;
- SetPictureInfo⁽³⁶³⁾ changes picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾;
- SetControllInfo⁽³⁵⁴⁾ changes inserted control⁽¹⁶⁸⁾;
- also
- SetItemTag⁽³⁶⁰⁾ changes item's "tag";
- SetItemVAlign⁽³⁶¹⁾ – change the item position relative to the line;
- SetItemText⁽³⁶⁰⁾ changes text/name of item;
- SetItemExtraIntProperty[Ex]⁽³⁵⁸⁾,
SetItemExtraStrProperty[Ex]⁽³⁵⁹⁾ sets value of extra item property;
- SetCheckpointInfo⁽³⁵³⁾ sets or changes item's "checkpoint";
- RemoveCheckpoint⁽³³²⁾ removes item's "checkpoint";
- see also properties PageBreaksBeforeItems⁽²⁴⁴⁾, ClearLeft⁽²²⁵⁾, ClearRight⁽²²⁶⁾.

Working with checkpoints

Obtaining checkpoint data:

- GetFirstCheckpoint⁽²⁹⁴⁾, GetNextCheckpoint⁽³⁰⁷⁾ allow to receive full list of *checkpoints*;
- GetLastCheckpoint⁽³⁰⁵⁾, GetPrevCheckpoint⁽³¹¹⁾ – the same in backward direction;
- GetCheckPointByNo⁽²⁹¹⁾ returns a *checkpoint* with the specified index;
- GetItemCheckpoint⁽²⁹⁸⁾ returns the item's *checkpoint*;
- event OnCheckPointVisible⁽³⁷²⁾ occurs when *checkpoint* (or "section", marked by this *checkpoint*) becomes visible as a result of vertical scrolling;
- FindCheckPointByName⁽²⁸⁷⁾ searches for *checkpoint* with the specified name;
- FindCheckPointByTag⁽²⁸⁷⁾ searches for *checkpoint* with the specified *tag*.

Extracting properties from *checkpoint* data:

- `GetCheckpointInfo`⁽²⁹¹⁾ returns main properties of the *checkpoint*;
- `GetCheckpointItemNo`⁽²⁹¹⁾ returns an index of item which owns the *checkpoint*;
- `GetCheckpointNo`⁽²⁹²⁾ returns an index of the *checkpoint*;
- `GetCheckpointYEx`⁽²⁹³⁾, `GetCheckpointXY`⁽²⁹²⁾ returns coordinates of the *checkpoint*.

Also:

- `GetCheckPointY`⁽²⁹²⁾ returns Y coordinate of *checkpoint* by its index.

See also methods for obtaining information about items⁽²¹⁵⁾ and for modifying document⁽²¹⁶⁾.

Mouse events

- `OnRVDbClick`⁽³⁹¹⁾ occurs on double click;
- `OnRVMouseDown`⁽³⁹⁴⁾ occurs when user presses mouse button;
- `OnRVMouseUp`⁽³⁹⁶⁾ occurs when user releases mouse button;
- `OnRVRightClick`⁽³⁹⁷⁾ occurs on right click.
- `OnItemHint`⁽³⁸¹⁾ allows customizing hints displayed when the mouse pointer moves above an item.
- `OnGetItemCursor`⁽³⁷⁴⁾ allows customizing cursors displayed when the mouse pointer moves above an item.

See also hypertext events⁽²¹²⁾.

Live spelling check

- `StartLiveSpelling`⁽³⁶⁷⁾ starts a live spelling thread;
- `ClearLiveSpellingResults`⁽²⁷⁹⁾ clears all live spelling underlines and stops the thread;
- `LiveSpellingValidateWord`⁽³¹⁸⁾ removes live spelling underlines from the specified word;
- `OnSpellingCheck`⁽⁴⁰⁹⁾ event – the place to check word spelling.

Picture animation**Properties:**

- `AnimationMode`⁽²²²⁾ specifies when image animation⁽¹¹⁹⁾ starts.

Methods:

- `StartAnimation`⁽³⁶⁶⁾ starts all animations;
- `StopAnimation`⁽³⁶⁷⁾ stops all animations;
- `ResetAnimation`⁽³³³⁾ rewinds all animations to the first frame.

Working in palette mode

- property `DoInPaletteMode`⁽²³⁴⁾;
- method `UpdatePaletteInfo`⁽³⁶⁸⁾.

Custom drawing**Events:**

- `OnPaint`⁽³⁸⁴⁾ – painting on top of content;
- `OnAfterDrawImage`⁽³⁷⁰⁾ – painting on top of images.

Others**Properties:**

- RVData⁽²⁴⁷⁾ – object representing document;
- Cursor⁽²²⁷⁾ – cursor;
- Delimiters⁽²²⁸⁾ – characters not belonging to words;
- TabNavigation⁽²⁵⁵⁾ sets mode for navigation through hypertext links and controls;
- DocObjects⁽²²⁹⁾ – additional object information for saving in RVF⁽¹²⁴⁾;
- DocProperties⁽²³¹⁾ – additional text information for saving in RVF⁽¹²⁴⁾;
- DocParameters⁽²³⁰⁾ – store document page parameters;
- NoteText⁽²⁴⁰⁾ – text for references to parent notes;
- UseVCLThemes⁽²⁵⁷⁾ [VCL] allows using colors of VCL themes (in Delphi XE2+);
- UseFMXThemes⁽²⁵⁶⁾ [FMX] allows using text color of FMX style
- DocumentPixelsPerInch⁽²³³⁾ [VCL,LCL] / ZoomPercent⁽²⁵⁸⁾ [FMX] allows zooming.

Methods:

- DeleteUnusedStyles⁽²⁸⁶⁾ deletes unused text, paragraph and list styles from TRVStyle;
- MarkStylesInUse⁽³³¹⁾, DeleteMarkedStyles⁽²⁸⁴⁾ – advanced methods for deleting unused styles;
- SearchText, -A, -W⁽³⁴⁶⁾ search for the text string; OnTextFound⁽⁴¹⁰⁾ event occurs when a text is found;
- StoreSearchResult⁽³⁶⁷⁾ allows storing a position of a search result (to continue search from this position);
- GetWordAt⁽³¹⁴⁾ returns a word at the specified point;
- DeleteItems⁽²⁸³⁾, DeleteSection⁽²⁸⁵⁾ delete the specified range of items from document;
- DeleteParas⁽²⁸⁵⁾ deletes the specified range of paragraphs, updates the document;
- FindControlItemNo⁽²⁸⁷⁾ returns index of item which owns the specified inserted control⁽¹⁶⁸⁾;
- GetFocusedItem⁽²⁹⁵⁾ returns the item in the focus;
- GetLineNo⁽³⁰⁶⁾ returns line number in the specified position;
- BeginOleDrag⁽²⁷⁸⁾ starts drag&drop.

Events:

- OnItemAction⁽³⁸⁰⁾;
- OnControlAction⁽³⁷³⁾.

6.1.1.1 TRichView: component editor

Component editor for TCustomRichView

(and all inherited components: TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾).

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`⁽²⁵¹⁾, `RVFParaStylesReadMode`⁽²⁵¹⁾, `RVFOptions`⁽²⁴⁷⁾, `RTFReadProperties.TextStyleMode`⁽⁴⁵⁸⁾, `.ParaStyleMode`⁽⁴⁵⁶⁾, `UseStyleTemplates`⁽²⁵⁶⁾. 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⁽⁶⁵⁴⁾, paragraph⁽⁶⁴⁸⁾ and list⁽⁶⁴⁶⁾ styles of the linked⁽²⁵³⁾ `TRVStyle`⁽⁶³⁰⁾ component;
- does not save these styles in RVF⁽¹²⁴⁾ files or streams.

"Allow adding styles dynamically" mode

In this mode, the component:

- adds new items in the collection of text⁽⁶⁵⁴⁾, paragraph⁽⁶⁴⁸⁾ and list⁽⁶⁴⁶⁾ styles, if necessary; new styles can be added as a result of RVF, RTF, DocX loading;
- saves these styles in RVF⁽¹²⁴⁾ 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`⁽⁵⁸⁵⁾ or `OnParaStyleConversion`⁽⁵⁸²⁾, 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`⁽²⁵⁶⁾ = `True`.

Additional information about these modes can be found in the topic "Styles and style templates"⁽⁹⁸⁾.

RVF⁽¹²⁴⁾

Defines options for saving and loading RVF (RichView Format)⁽¹²⁴⁾ files and strings. These settings change `RVFOptions`⁽²⁴⁷⁾.

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`

- `AllowSelection`⁽²²²⁾ (deprecated)
- `AnimationMode`⁽²²²⁾
- `BackgroundBitmap`⁽²²²⁾
- `BackgroundStyle`⁽²²³⁾
- `BiDiMode`⁽²²⁴⁾
- `BottomMargin`⁽²²⁵⁾
- `ClearLeft`⁽²²⁵⁾

- ClearRight ⁽²²⁶⁾
- Color ⁽²²⁶⁾
- CPEventKind ⁽²²⁷⁾
- Cursor ⁽²²⁷⁾
- DarkMode ⁽²²⁸⁾
- Delimiters ⁽²²⁸⁾
- DocObjects ⁽²²⁹⁾
- DocParameters ⁽²³⁰⁾
- DocProperties ⁽²³¹⁾
- Document ⁽²³²⁾
- ▶ DocumentHeight ⁽²³³⁾
- DocumentPixelsPerInch ⁽²³³⁾ [VCL,LCL]
- DoInPaletteMode ⁽²³⁴⁾
- ▶ FirstItemVisible ⁽²³⁶⁾
- FirstJumpNo ⁽²³⁶⁾
- HTMLReadProperties ⁽²³⁷⁾
- HTMLSaveProperties ⁽²³⁷⁾
- ▶ ItemCount ⁽²³⁷⁾
- ▶ LastItemVisible ⁽²³⁸⁾
- LeftMargin ⁽²³⁸⁾
- MarkdownProperties ⁽²³⁹⁾
- MaxLength ⁽²³⁹⁾
- MaxTextWidth ⁽²³⁹⁾
- MinTextWidth ⁽²⁴⁰⁾
- NoteText ⁽²⁴⁰⁾
- Options ⁽²⁴⁰⁾
- PageBreaksBeforeItems ⁽²⁴⁴⁾
- RightMargin ⁽²⁴⁴⁾
- RTFOptions ⁽²⁴⁵⁾
- RTFReadProperties ⁽²⁴⁶⁾
- ▶ RVData ⁽²⁴⁷⁾
- RVFOptions ⁽²⁴⁷⁾
- RVFParaStylesReadMode ⁽²⁵¹⁾
- RVFTextStylesReadMode ⁽²⁵¹⁾
- ▶ RVFWarnings ⁽²⁵¹⁾
- SingleClick ⁽²⁵³⁾ (deprecated)
- SelectionHandlesVisible ⁽²⁵³⁾ (D2010+)
- **Style** ⁽²⁵³⁾
- StyleTemplateInsertMode ⁽²⁵⁴⁾
- TabNavigation ⁽²⁵⁵⁾
- TopMargin ⁽²⁵⁵⁾
- UseFMXThemes ⁽²⁵⁷⁾ [FMX]
- UseStyleTemplates ⁽²⁵⁶⁾
- UseVCLThemes ⁽²⁵⁷⁾ [VCL]
- VAlign ⁽²⁵⁷⁾
- VSmallStep ⁽²⁵⁷⁾ [VCL,LCL]
- WordWrap ⁽²⁵⁸⁾
- ZoomPercent ⁽²⁵⁸⁾ [FMX]

Derived from TRVScroller ⁸¹³

- BorderStyle ⁸¹⁶ [VCL]
- ▶ HScrollMax ⁸¹⁶
- HScrollPos ⁸¹⁶
- HScrollVisible ⁸¹⁶ [VCL,LCL]
- ▶ InplaceEditor ⁸¹⁷
- NoVScroll ⁸¹⁷
- Tracking ⁸¹⁸
- UseXPThemes ⁸¹⁸ [VCL,LCL]
- ▶ VScrollMax ⁸¹⁸
- VScrollPos ⁸¹⁹
- VScrollVisible ⁸¹⁹ [VCL,LCL]
- WheelStep ⁸¹⁹ [VCL,LCL]

**Derived from TCustomControl [VCL/LCL]
or TCustomScrollBar [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
- ShowScrollBar [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 ⁽²⁴⁰⁾

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
<i>rvaniDisabled</i>	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.
<i>rvaniManualStart</i>	Animation starts when calling StartAnimation ⁽³⁶⁶⁾ and stops when calling StopAnimation ⁽³⁶⁷⁾ .
<i>rvaniOnFormat</i>	Animation starts when calling Format ⁽²⁸⁸⁾ and stops when calling Clear ⁽²⁷⁹⁾ . StartAnimation ⁽³⁶⁶⁾ and StopAnimation ⁽³⁶⁷⁾ work as well.

Default value:

rvaniManualStart

See also:

- Animation in TRichView ⁽¹¹⁹⁾;
- Pictures ⁽¹⁶³⁾;
- Hot-Pictures ⁽¹⁶⁶⁾.

6.1.1.2.3 TRichView.BackgroundBitmap

Background image shown below the document

property BackgroundBitmap: TBitmap;

Assignment to this property copies the image, so you still need to free the source image yourself.

Example 1 (assigning image)

```

bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
MyRichView.BackgroundBitmap := bmp;
bmp.Free;

```

Example 2 (doing the same work)

```

MyRichView.BackgroundBitmap.LoadFromFile('c:\image.bmp');

```

Example 3 (clearing the image)

```

MyRichView.BackgroundBitmap := nil;

```

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetBackgroundImageEd`⁽⁵⁴⁶⁾.

See also properties:

- `BackgroundStyle`⁽²²³⁾.

6.1.1.2.4 TRichView.BackgroundStyle

Defines how `BackgroundBitmap`⁽²²²⁾ is displayed

type

```

TBackgroundStyle = (bsNoBitmap, bsStretched,
  bsTiled, bsTiledAndScrolled, bsCentered,
  bsTopLeft, bsTopRight,
  bsBottomLeft, bsBottomRight);

```

property `BackgroundStyle` : `TBackgroundStyle`;

Note: `TBackgroundStyle` is declared in `RVScroll` unit.

Style	Meaning
<i>bsNoBitmap</i>	Colored background (bitmap is not used)
<i>bsStretched</i>	Bitmap is stretched to fit RichView window
<i>bsTiled</i>	Repeats bitmap until the entire RichView window is covered
<i>bsTiledAndScrolled</i>	Repeats bitmap until the entire scrollable area is covered (tiles are scrolled together with content)
<i>bsCentered</i>	Centers bitmap in RichView window
<i>bsTopLeft</i> ,	Bitmap is in the top left corner
<i>bsTopRight</i>	Bitmap is in the top right corner
<i>bsBottomLeft</i>	Bitmap is in the bottom left corner
<i>bsBottomRight</i>	Bitmap is in the bottom right corner

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetIntPropertyEd`⁽⁵⁴⁵⁾.

See also properties:

- BackgroundBitmap⁽²²²⁾;
- Color⁽²²⁶⁾.

See also:

- TRVItemBackgroundType⁽¹⁰¹³⁾ type;
- Smooth image scaling⁽¹²⁰⁾.

6.1.1.2.5 TRichView.BiDiMode

Specifies the default bidirectional mode for the control.

property BiDiMode: TRVBiDiMode⁽⁹⁹³⁾;

(introduced in version 1.6)

Value	Meaning
<i>rvbdUnspecified</i>	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.
<i>rvbdLeftToRight</i>	Sets default text flow to left-to-right
<i>rvbdRightToLeft</i>	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⁽⁷²¹⁾ and BiDiMode of text style⁽⁶⁹⁹⁾.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

The results depends on the value of Style⁽²⁵³⁾.TextEngine⁽⁶⁵⁴⁾:

- 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⁽¹³²⁾.

Default value:

rvbdUnspecified

6.1.1.2.6 TRichView.BottomMargin

Bottom margin of document

property BottomMargin: TRVPixelLength⁽¹⁰¹⁹⁾;

This value is measured in "standard" pixels, 1 "standard" pixel = 1/Style⁽²⁵³⁾.UnitsPerInch⁽⁶⁶⁵⁾ of an inch. When drawing on a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

See also properties:

- LeftMargin⁽²³⁸⁾;
- RightMargin⁽²⁴⁴⁾;
- TopMargin⁽²⁵⁵⁾.

See also properties of TRVPrint:

- Margins⁽⁷⁶⁸⁾.

See also:

Additional information about horizontal scrolling, margins, indents, etc.⁽¹⁰⁵⁾

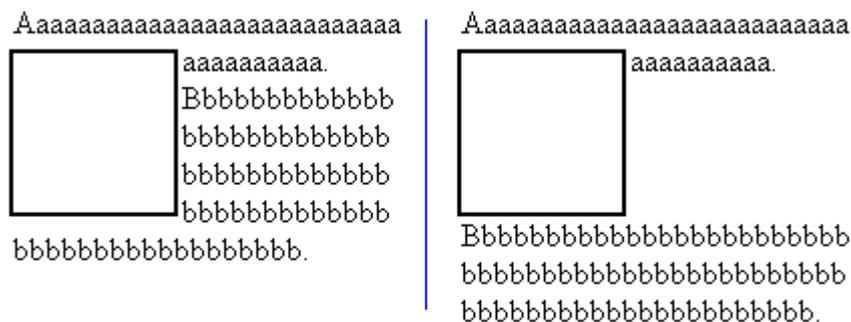
6.1.1.2.7 TRichView.ClearLeft

Specifies whether a text flow around left-aligned⁽¹⁰³³⁾ 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⁽¹⁰³³⁾ 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⁽³¹⁸⁾(ItemNo)) or a new line inside a paragraph (IsFromNewLine⁽³¹⁷⁾(ItemNo)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using Format⁽²⁸⁸⁾).

In editor you can change value of this property as an editing operation, using ClearTextFlow⁽⁵⁰¹⁾ method.

Note: if special characters are shown (i.e. *rvShowSpecialCharacters* is included in Options⁽²⁴⁰⁾), 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*⁽²²⁶⁾,
- *PageBreaksBeforeItems*⁽²⁴⁴⁾,
- *TRVAlign*⁽¹⁰³³⁾ type (*rvvaLeft*).

6.1.1.2.8 TRichView.ClearRight

Specifies whether a text flow around right-aligned⁽¹⁰³³⁾ 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⁽¹⁰³³⁾ object.

This item must start a new paragraph (`IsParaStart`⁽³¹⁸⁾(`ItemNo`)) or a new line inside a paragraph (`IsFromNewLine`⁽³¹⁷⁾(`ItemNo`)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using `Format`⁽²⁸⁸⁾).

In editor you can change value of this property as an editing operation, using `ClearTextFlow`⁽⁵⁰¹⁾ method.

Note: if special characters are shown (i.e. *rvShowSpecialCharacters* is included in Options⁽²⁴⁰⁾), 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*⁽²²⁵⁾,
- *PageBreaksBeforeItems*⁽²⁴⁴⁾,
- *TRVAlign*⁽¹⁰³³⁾ type (*rvvaRight*).

6.1.1.2.9 TRichView.Color

Background color

property `Color: TRVColor`⁽⁹⁹⁶⁾;

This property allows to override the default background color specified in `RichView.Style`⁽²⁵³⁾.`Color`⁽⁶³⁵⁾.

`TRichView` paints background with this color, if it is not equal to `rvcNone`⁽¹⁰³⁸⁾, otherwise `TRichView` uses the default color.

Background color is used if `BackgroundBitmap`⁽²²²⁾ is empty or does not cover the entire window (see `BackgroundStyle`⁽²²³⁾).

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetIntPropertyEd`⁽⁵⁴⁵⁾.

Default value:

rvclNone⁽¹⁰³⁸⁾

See also properties:

- *BackgroundStyle*⁽²²³⁾;
- *BackgroundBitmap*⁽²²²⁾.

See also properties of TRVStyle:

- *Color*⁽⁶³⁵⁾.

6.1.1.2.10 TRichView.CPEventKind

Specifies the mode of generating *OnCheckpointVisible*⁽³⁷²⁾ event.

type

```
TCPEventKind =
  (cpeNone, cpeAsSectionStart, cpeWhenVisible);
```

property *CPEventKind*: TCPEventKind;

Note: TCPEventKind is declared in RVScroll unit.

Value	Meaning
<i>cpeNone</i>	<i>OnCheckpointVisible</i> ⁽³⁷²⁾ does not occur
<i>cpeAsSectionStart</i>	<i>OnCheckpointVisible</i> ⁽³⁷²⁾ is called for visible <i>checkpoint</i> or the nearest <i>checkpoint</i> above the visible area (so that the last event is always called for <i>checkpoint</i> located in or above the visible area)
<i>cpeWhenVisible</i>	<i>OnCheckpointVisible</i> ⁽³⁷²⁾ is generated for visible <i>checkpoint</i>

This event occurs only for *checkpoints* having "raise event" flag.

This event never occurs in TRichViewEdit (in this version)

See also events:

- *OnCheckpointVisible*⁽³⁷²⁾.

See also:

- *Checkpoints*⁽⁸⁷⁾ for details.

6.1.1.2.11 TRichView.Cursor

Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.

property *Cursor*: TCursor;

Default value:

- *crDefault* for TRichView;
- *crIBeam* for TRichViewEdit⁽⁴⁶¹⁾.

See also:

- RichViews' cursors ¹⁴².

6.1.1.2.12 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 ²²⁶ and Style ²⁵³.Color ⁶³⁵ 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 ²⁵⁷
- UseFMXThemes ²⁵⁶
- TCustomRVPrint ⁸²².DarkMode ⁸²³

6.1.1.2.13 TRichView.Delimiters

Set of delimiters for specifying a "word" in document

property Delimiters: TRVUnicodeString ¹⁰³²;

(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 ³⁴⁹;
- GetWordAt ³¹⁴;
- SearchText ³⁴⁶.

See also events:

- OnRVDbClick⁽³⁹¹⁾;
- OnRVRightClick⁽³⁹⁷⁾.

See also methods of TRichViewEdit:

- SelectCurrentWord⁽⁵⁴⁵⁾.

See also:

- Live spelling check⁽¹³²⁾.

6.1.1.2.14 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⁽²⁴⁷⁾, these objects are saved in RVF.

If *rvfoLoadDocObjects* is in RVFOptions⁽²⁴⁷⁾, methods for loading RVF load these objects. Methods for inserting RVF ignore these objects.

Method TRichView.Clear⁽²⁷⁹⁾ clears **DocObjects**.

There is one more property allowing to save custom data in RVF: DocProperties⁽²³¹⁾.

Example 1

TRVMathDocObject⁽⁹⁷⁴⁾ defines default properties for items displaying mathematical expressions⁽¹⁸⁷⁾.

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 ⁽²⁴⁷⁾.

How to load from RVF: include *rvfoLoadDocObjects* in RVFOptions ⁽²⁴⁷⁾.

See also methods for saving RVF (they can save this property):

- SaveRVF ⁽³⁴⁴⁾;
- SaveRVFToStream ⁽³⁴⁴⁾.

See also methods for loading RVF (they can load this property):

- LoadRVF ⁽³²⁸⁾;
- LoadRVFFromStream ⁽³²⁹⁾.

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd ⁽⁵²⁸⁾;
- InsertRVFFromStreamEd ⁽⁵²⁸⁾;
- PasteRVF ⁽⁵³⁸⁾.

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream ⁽³¹⁶⁾.
- AppendRVFFromStream ⁽²⁷⁶⁾.

See also properties:

- RVFOptions ⁽²⁴⁷⁾.

See also:

- Saving and loading RichView documents ⁽¹²²⁾;
- RVF ⁽¹²⁴⁾.

6.1.1.2.15 TRichView.DocParameters

Page properties.

```
property DocParameters: TRVDocParameters (415);
```

(introduced in version 10)

Subproperties of this property are not used by TRichView itself, but can be written and read in RVF ⁽¹²⁴⁾, RTF, DocX, HTML.

Besides, they can be assigned to TRVPrint ⁽⁷⁵⁹⁾ using the method AssignDocParameters ⁽⁷⁷¹⁾.

RVF

DocParameters are stored with DocProperties⁽²³¹⁾.

How to save to RVF: include *rvfoSaveDocProperties* in RVFOptions⁽²⁴⁷⁾.

How to load from RVF: include *rvfoLoadDocProperties* in RVFOptions⁽²⁴⁷⁾.

RTF and DocX

How to save to RTF/DocX: include *rvrtfSaveDocParameters* in RTFOptions⁽²⁴⁵⁾.

How to load from RTF/DocX: set RTFReadProperties⁽²⁴⁶⁾.ReadDocParameters⁽⁴⁵⁷⁾ = *True*.

HTML

Title⁽⁴¹⁹⁾ and Comments⁽⁴¹⁷⁾ 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⁽²³⁷⁾.ReadDocParameters⁽⁴²⁷⁾ = *True*.

See also events:

- OnSaveRTFExtra⁽⁴⁰⁶⁾;
- OnSaveDocXExtra⁽³⁹⁹⁾;
- OnSaveHTMLExtra⁽⁴⁰¹⁾.

6.1.1.2.16 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⁽²⁴⁷⁾, these strings are saved in RVF.

If *rvfoLoadDocProperties* is in RVFOptions⁽²⁴⁷⁾, 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⁽²²⁹⁾.

Method TRichView.Clear⁽²⁷⁹⁾ clears DocProperties.

See also methods for saving RVF (they can save this property):

- SaveRVF⁽³⁴⁴⁾;
- SaveRVFtoStream⁽³⁴⁴⁾.

See also methods for loading RVF (they can load this property):

- LoadRVF⁽³²⁸⁾;
- LoadRVFFromStream⁽³²⁹⁾.

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd⁽⁵²⁸⁾;
- InsertRVFFromStreamEd⁽⁵²⁸⁾;

- PasteRVF⁽⁵³⁸⁾.

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream⁽³¹⁶⁾;
- AppendRVFFromStream⁽²⁷⁶⁾.

See also properties:

- RVFOptions⁽²⁴⁷⁾.

See also:

- Saving and loading RichView documents⁽¹²²⁾;
- RVF⁽¹²⁴⁾.

6.1.1.2.17 TRichView.Document

Allows linking this TRichView component to a database field (Delphi XE2+)

property Document: TRVDocumentProperty⁽⁴²²⁾;

(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⁽⁵⁹⁴⁾ and TDBRichViewEdit⁽⁶⁰⁶⁾ 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⁽¹²⁴⁾). After copying, the destination TRichView control is formatted (unless it is TRVReportHelper's RichView).

See also events:

- OnNewDocument⁽³⁸⁴⁾
- OnLoadDocument⁽³⁸⁴⁾
- OnLoadCustomFormat⁽³⁸³⁾

See also events of TRichViewEdit⁽⁴⁶¹⁾:

- OnSaveCustomFormat⁽⁵⁸⁵⁾

6.1.1.2.18 TRichView.DocumentHeight

Height of document (height of scrollable area)

property DocumentHeight: TRVCoord⁽²³³⁾;

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⁽¹⁰⁵⁾.

6.1.1.2.19 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⁽³¹¹⁾ 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⁽²⁶⁸⁾ or Reformat⁽³³²⁾.

Zooming is applied to almost everything: margins, background images, items in a document. However, there are exceptions; the following items are not affected:

- controls⁽¹⁶⁸⁾
- images from image lists (*bullets*⁽¹⁷⁰⁾ and *hot-spots*⁽¹⁷²⁾)*

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(311) /
    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⁽⁶³⁰⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾
- ZoomPercent⁽²⁵⁸⁾ (FireMonkey analog)

6.1.1.2.20 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
<i>rvpaDoNothing</i>	RichView does nothing with palette. Use if you have no pictures in RichView, or all pictures are in VGA 16-color palette.
<i>rvpaAssignPalette</i>	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

Mode	Meaning
	use this option if you want only to display document.
<i>rvpaCreateCopies</i>	(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.
<i>rvpaCreateCopiesEx</i>	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:

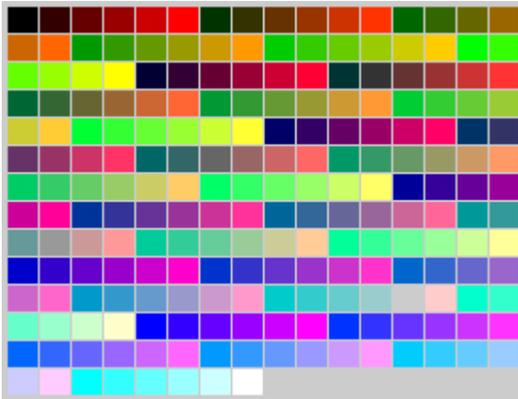
```
procedure TMyForm.WMDisplayChange(var Message: TWMDisplayChange);
begin
  // Call UpdatePaletteInfo methods of all your
  // RichViews, RichViewEdits, and RVPrints (if you use print preview)
  // for example:
  MyRichViewEdit.UpdatePaletteInfo(368);
  MyRVPrint.UpdatePaletteInfo(828);
end;
```

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`⁽³⁶⁸⁾.

See also methods of TRVPrint:

- `UpdatePaletteInfo`⁽⁸²⁸⁾.

6.1.1.2.21 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`⁽⁸²¹⁾ 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`⁽²³⁸⁾.

See also:

- Vertical scrolling⁽¹⁰⁵⁾.

6.1.1.2.22 TRichView.FirstJumpNo

Hypertext **id** for the first hyperlink in the document.

property `FirstJumpNo`: `Integer`;

When you append hypertext items (*hotspots*⁽¹⁷²⁾, *hot-pictures*⁽¹⁶⁶⁾ 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 **ids**. **Ids** are incremented according to their order in the document, from top to bottom.

This property is useful if you want to use one OnJump³⁸² event handler for several RichViews.

Default value:

0

See also:

- RichView hypertext⁹².

6.1.1.2.23 TRichView.HTMLReadProperties

A set of properties controlling HTML import.

```
property HTMLReadProperties: TRVHTMLReaderProperties424;
```

(introduced in version 21)

These properties are used by:

- TRichView.LoadHTML³²², LoadHTMLFromStream³²⁴
- TRichViewEdit.InsertHTMLFromFileEd⁵²¹, InsertHTMLFromStreamEd⁵²¹;
- TRichViewEdit.PasteHTML⁵³⁶ (and Paste⁵³⁴);
- TRichView.LoadFromFile³²⁰, LoadFromStream³²¹.

Read the topic about TRVHTMLReaderProperties⁴²⁴ for details.

See also:

- HTMLSaveProperties²³⁷

6.1.1.2.24 TRichView.HTMLSaveProperties

A set of properties controlling HTML export.

```
property HTMLSaveProperties: TRVHTMLSaveProperties429;
```

(introduced in version 21)

These properties are used by SaveHTML³³⁵ and SaveHTMLToStream³³⁸ methods.

Read the topic about TRVHTMLSaveProperties⁴²⁹ for details.

See also:

- HTMLReadProperties²³⁷

6.1.1.2.25 TRichView.ItemCount

Number of items in RichView

```
property ItemCount: Integer;
```

See also:

- Item types⁸⁵;
- Adding items¹⁰²;
- Obtaining items¹¹¹;
- Modifying items¹¹².

6.1.1.2.26 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⁽⁸²¹⁾ 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⁽²³⁶⁾.

See also:

- Vertical scrolling⁽¹⁰⁵⁾.

6.1.1.2.27 TRichView.LeftMargin

Left margin of document

property LeftMargin: TRVPixelLength⁽¹⁰¹⁹⁾;

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style⁽²⁵³⁾.UnitsPerInch⁽⁶⁵⁵⁾ of an inch. When drawing on 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⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

See also properties:

- RightMargin⁽²⁴⁴⁾;
- TopMargin⁽²⁵⁵⁾;
- BottomMargin⁽²²⁵⁾.

See also indents in paragraph style⁽⁷¹⁸⁾.

See also properties of TRVPrint:

- Margins⁽⁷⁶⁸⁾.

See also properties of TRVStyle:

- LineSelectCursor⁽⁶⁴³⁾.

See also:

Additional information about horizontal scrolling, margins, indents, etc.⁽¹⁰⁵⁾

6.1.1.2.28 TRichView.LineCount

Obsolete, use ItemCount⁽²³⁷⁾.

property LineCount : Integer;

See also

- GetLineNo⁽³⁰⁶⁾.

6.1.1.2.29 TRichView.MarkdownProperties

A set of properties controlling **Markdown** export and import.

```
property MarkdownProperties: TRVMarkdownProperties439;
```

(introduced in version 19)

Read the topic about TRVMarkdownProperties⁴³⁹ for details.

Subproperties of this class affect the following methods:

- TRichView.SaveMarkdown³⁴⁰, SaveMarkdownToStream³⁴¹;
- TRichView.LoadMarkdown³²⁵, LoadMarkdownFromStream³²⁵;
- TRichView.LoadFromFile³²⁰, LoadFromStream³²¹;
- TRichViewEdit.InsertMarkdownFromFileEd⁵²³, InsertMarkdownFromStreamEd⁵²⁴.

6.1.1.2.30 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 WordWrap²⁵⁸ = *False*. This property is ignored for paragraphs having *rvpaoNoWrap* in Options⁷²³.

Default value:

0

6.1.1.2.31 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: TRVPixelLength1019;
```

This property does not affect printing.

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style²⁵³.UnitsPerInch⁶⁵⁵ of an inch. When drawing on a screen, this value is recalculated according to the screen DPI (dots per inch) value.

In TRichViewEdit⁴⁶¹, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁵⁴⁵.

Default value:

0

See horizontal scrolling¹⁰⁵ for details.

See also properties:

- MaxLength²³⁹.

6.1.1.2.32 TRichView.MinTextWidth

Minimum width for text wrapping. This value allows to increase width of scrollable area in TRichView.

property MinTextWidth : TRVPixelLength⁽¹⁰¹⁹⁾;

This property does not affect printing.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

Default value:

0

See horizontal scrolling⁽¹⁰⁵⁾ for details.

6.1.1.2.33 TRichView.NoteText

Text for references to the parent note⁽¹⁸⁴⁾.

property NoteText: TRVUnicodeString⁽¹⁰³²⁾;

(introduced in version 10)

Normally, references⁽¹⁸⁴⁾ must be placed in Document⁽⁹²⁶⁾ for footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁸⁰⁾ or sidenotes⁽¹⁸¹⁾. In this case, they display the same text as their parent note.

References⁽¹⁸⁴⁾ 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⁽⁹²⁶⁾ for some note, you should set value of this property equal to NoteText⁽⁹²⁷⁾ of this note.

Default value:

'?'

6.1.1.2.34 TRichView.Options

Options

type

```
TRVOption = (rvoAllowSelection, rvoSingleClick, rvoScrollToEnd,
  rvoClientTextWidth,
  rvoShowCheckpoints, rvoShowPageBreaks,
  rvoShowSpecialCharacters, rvoShowHiddenText,
  rvoTagsArePChars,
  rvoAutoCopyText, rvoAutoCopyUnicodeText
  rvoAutoCopyRVF, rvoAutoCopyImage,
  rvoAutoCopyRTF,
  rvoFormatInvalidate,
  rvoDb1ClickSelectsWord, 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*, *rvoDbClickSelectsWord*, *rvoRClickDeselects*, *rvoFormatInvalidate*, *rvoShowPageBreaks*, *rvoShowGridLines*, *rvoFastFormatting*]

For FireMonkey: the same + *rvoDefaultSelectionFill*.

For older versions: the same, but with *rvoAutoCopyText* instead of *rvoAutoCopyUnicodeText*.

General Options

Option	Meaning
<i>rvoTagsArePChars</i>	Obsolete, does nothing.
<i>rvoClientTextWidth</i>	If set, text always wraps to fit client width of RichView. Also affects breaks ⁽¹⁶⁷⁾ and tables ⁽¹⁷³⁾ . See details ⁽¹⁰⁵⁾ . Can be changed: any time, but you need to reformat ⁽³³²⁾ and repaint RichView after changing this option, if the document was already formatted.
<i>rvoScrollToEnd</i>	If set (default), RichView scrolls to the end of document when FormatTail ⁽²⁸⁸⁾ method is called. If cleared, document does not scroll in this case Can be changed: any time.
<i>rvoSingleClick</i>	If set, OnRVDbClick ⁽³⁹¹⁾ event is generated on single mouse click If cleared (default), it is generated on double mouse click Can be changed: any time.
<i>rvoFormatInvalidate</i>	If set (default), Format ⁽²⁸⁸⁾ and FormatTail ⁽²⁸⁸⁾ methods repaint component. Can be changed: any time.
<i>rvoDisallowDrag</i>	If set, dragging ⁽¹¹⁸⁾ from this TRichView is not allowed. Can be changed: any time.
<i>rvoFastFormatting</i>	Increase performance in WinNT-based OS at the cost of some resources
<i>rvoPercentEncodedURL</i>	If set, it is assumed that URLs (stored in tags or provided in OnWriteHyperlink ⁽⁴¹¹⁾ 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.

Option	Meaning
<i>rvoAssignImageFileNames</i>	<p>If set, the component assigns file names of images, when possible.</p> <p>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).</p> <p>The following properties are set:</p> <ul style="list-style-type: none"> • <code>table.BackgroundImageFileName</code>⁽⁸⁵⁰⁾ • <code>cell.BackgroundImageFileName</code>⁽⁸⁹⁷⁾ • <code>rvesplImageFileName</code>⁽¹⁰⁰⁵⁾ item property <p>See also: <code>OnAssignImageFileName</code>⁽³⁷¹⁾ event.</p>

Selection Options

Can be changed: any time.

Option	Meaning
<i>rvoAllowSelection</i>	<p>If set (default), user can select content of RichView using mouse or keyboard.</p> <p>Please do not clear this option for TRichViewEditor! See: working with selection⁽¹⁰⁷⁾.</p>
<i>rvoDbClickSelectsWord</i>	<p>If set (default), double click on word selects it (and double click on non-text item selects this item)</p>
<i>rvoRClickDeselects</i>	<p>If set (default), right click outside the selected area deselects (and moves caret to the clicked position in TRichViewEdit⁽⁴⁶¹⁾)</p>

Show/Hide Options

Option	Meaning
<i>rvoShowCheckpoints</i>	<p>If set, all <i>checkpoints</i> are shown (as dotted horizontal lines, or by <code>OnDrawCheckpoint</code>⁽⁶⁶⁷⁾). See details⁽⁸⁷⁾.</p> <p>Can be changed: any time, you need to repaint RichView after changing this option</p>
<i>rvoShowPageBreaks</i>	<p>If set (default), page breaks are shown as horizontal lines.</p> <p>Can be changed: any time, you need to repaint RichView after changing this option</p>
<i>rvoShowSpecialCharacters</i>	<p>If set, nonprinting characters are displayed as special marks: spaces, nonbreaking spaces, paragraph breaks,</p>

Option	Meaning
	line breaks, tabs ⁽¹⁶²⁾ , soft hyphens in Unicode text, lines to left- and right-aligned items, places of insertion of text boxes ⁽¹⁸³⁾ . Can be changed: any time, you need to reformat ⁽³³²⁾ RichView after changing this option, if the document is already formatted. See also: RVVisibleSpecialCharacters ⁽¹⁰⁶¹⁾ , RichViewShowGhostSpaces ⁽¹⁰⁴⁸⁾ typed constants.
<i>rvoShowHiddenText</i>	If set, hidden items ⁽¹⁰¹⁾ are displayed. Can be changed: any time, you need to reformat ⁽³³²⁾ RichView after changing this option, if the document is already formatted.
<i>rvoShowItemHints</i>	If set, items' hints are shown. Popup hints are shown only if ShowHints property is set to True. See item properties ⁽¹⁰⁰⁵⁾ and OnItemHint ⁽³⁸¹⁾ event.
<i>rvoShowGridLines</i>	If set, table ⁽¹⁷³⁾ grid lines ⁽⁶⁴⁰⁾ are shown.

Clipboard Options (See Details⁽¹⁰⁸⁾)

[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⁽²⁸¹⁾. 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
<i>rvoAutoCopyText</i>	ANSI text (only for VCL and LCL)
<i>rvoAutoCopyUnicodeText</i>	Unicode text
<i>rvoAutoCopyRVF</i>	RVF ⁽¹²⁴⁾ (RichView Format)
<i>rvoAutoCopyRTF</i>	RTF (Rich Text Format)
<i>rvoAutoCopyImage</i>	Bitmap or metafile (if the image is selected)

FireMonkey-specific Options

Option	Format to Copy
<i>rvoDefaultSelectionFill</i>	If set, and if 'selection' brush resource can be found in the linked visual style (specified in StyleLookup property), SelColor, SelOpacity, InactiveSelColor ⁽⁶⁴¹⁾ properties of the linked ⁽²⁵³⁾ TRVStyle ⁽⁶³⁰⁾ component are updated according to this resource.

6.1.1.2.35 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⁽²²⁵⁾[ItemNo] and ClearRight⁽²²⁶⁾[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⁽²⁴⁰⁾ (*rvoShowPageBreaks*).

See also methods:

- AssignSoftPageBreaks⁽²⁷⁷⁾.

See also properties and events of TRVStyle:

- PageBreakColor⁽⁶⁴⁷⁾;
- OnDrawPageBreak⁽⁶⁶⁸⁾.

See also methods of TRichViewEdit:

- InsertPageBreak⁽⁵²⁵⁾;
- RemoveCurrentPageBreak⁽⁵⁴²⁾.

See also properties of table rows:

- PageBreakBefore⁽⁹¹⁰⁾.

6.1.1.2.36 TRichView.RightMargin

Right margin of document

property RightMargin: TRVPixelLength⁽¹⁰¹⁹⁾;

This value is measured in “standard” pixels, 1 “standard” pixel = 1/Style⁽²⁵³⁾.UnitsPerInch⁽⁶⁵⁵⁾ of an inch. When drawing on a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

See also properties:

- LeftMargin⁽²³⁸⁾;
- TopMargin⁽²⁵⁵⁾;
- BottomMargin⁽²²⁵⁾.

See also indents in paragraph style⁽⁷¹⁸⁾

See also properties of TRVPrint:

- Margins⁽⁷⁶⁸⁾.

See also properties of TRVStyle:

- LineSelectCursor⁽⁶⁴³⁾.

See also:

Additional information about horizontal scrolling, margins, indents, etc.⁽¹⁰⁵⁾

6.1.1.2.37 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
<i>rvrtfSaveStyleSheet</i>	Used only if UseStyleTemplates ⁽²⁵⁶⁾ = <i>False</i> . If set (not recommended), RichView saves a style sheet in RTF; this style sheet is created from text and paragraph styles having Standard ⁽⁶⁹⁵⁾ = <i>True</i> . If UseStyleTemplates ⁽²⁵⁶⁾ = <i>True</i> , the component always saves Style ⁽²⁵³⁾ .StyleTemplates ⁽⁶⁵²⁾ as a style sheet.
<i>rvrtfDuplicateUnicode</i>	If set, RichView saves an ANSI equivalent of Unicode text ⁽¹³⁰⁾ . A conversion is based on CharSet ⁽⁷⁰⁰⁾ 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.
<i>rvrtfSaveEMFAsWMF</i>	If set, RichView saves EMF (enhanced (Win32) metafiles) as WMF (Windows (Win3.1) metafiles). Some RTF readers do not understand EMF in RTF.
<i>rvrtfSaveJpegAsJpeg</i>	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.
<i>rvrtfSavePngAsPng</i>	If set, RichView saves PNG images (PNG class is registered with RVGraphicHandler ⁽¹⁰⁵⁷⁾ .RegisterPngGraphic) as PNG. Otherwise, RichView saves PNG images in other formats (see below).

Option	Meaning
	Some RTF readers do not understand PNG images in RTF.
<i>rvrtfSaveBitmapDefault</i>	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 <i>rvrtfPNGInsteadOfBitmap</i> below. (unfortunately, many RTF reader do not understand WMF with mapping mode = MM_TEXT, which RichView uses for saving other formats; some other RTF readers do not understand bitmaps).
<i>rvrtfSaveEMFDefault</i>	If set (and <i>rvrtfSaveBitmapDefault</i> is not set), RichView saves graphics formats other than bitmaps (and optionally Jpegs) as EMF. If not set, as WMF
<i>rvrtfSavePicturesBinary</i>	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)
<i>rvrtfPNGInsteadOfBitmap</i>	If included, and PNG class is registered with RVGraphicHandler ⁽¹⁰⁵⁷⁾ .RegisterPngGraphic, bitmaps are saved as PNG (including saving of non-RTF picture types, if <i>rvrtfSaveBitmapDefault</i> is included). Not all RTF readers understand PNG.

Options for RTF and DocX

Option	Meaning
<i>rvrtfSaveDocParameters</i>	If included, properties of DocParameters ⁽²³⁰⁾ are saved in RTF and DocX.
<i>rvrtfSaveHeaderFooter</i>	If included, documents assigned by SetHeader ⁽³⁵⁶⁾ and .SetFooter ⁽³⁵⁵⁾ are saved as a header and footer in RTF and DocX.

Default value

[*rvrtfDuplicateUnicode*, *rvrtfSaveEMFAsWMF*, *rvrtfSaveJpegAsJpeg*, *rvrtfSavePngAsPng*]

See also methods:

- SaveRTF⁽³⁴³⁾;
- SaveRTFToStream⁽³⁴⁴⁾;
- CopyRTF⁽²⁸²⁾.

See also:

- Saving and loading in TRichView⁽¹²²⁾;
- Pictures⁽¹⁶³⁾.

6.1.1.2.38 TRichView.RTFReadProperties

A set of properties controlling RTF (Rich Text Format) and DocX (Microsoft Word Document) import.

property RTFReadProperties: TRVRTFReaderProperties⁽⁴⁵⁰⁾;

Read the topic about TRVRTFReaderProperties⁽⁴⁵⁰⁾ for details.

Subproperties of this class affect the following methods:

- TRichView.LoadRTF⁽³²⁶⁾, LoadDocX⁽³¹⁸⁾;
- TRichView.LoadRTFFromStream⁽³²⁷⁾, LoadDocXFromStream⁽³¹⁹⁾;
- TRichView.LoadFromFile⁽³²⁰⁾, LoadFromStream⁽³²¹⁾;
- TRichViewEdit.PasteRTF⁽⁵³⁷⁾ (and Paste⁽⁵³⁴⁾);
- TRichViewEdit.InsertRTFFromStreamEd⁽⁵²⁸⁾, InsertDocXFromStreamEd⁽⁵¹⁸⁾;
- TRichViewEdit.InsertRTFFromFileEd⁽⁵²⁸⁾, InsertDocXFromFileEd⁽⁵¹⁸⁾;
- drag&drop⁽¹¹⁸⁾ when dropping RTF file.

See also:

- RTFOptions⁽²⁴⁵⁾ (options for RTF and DocX export).

6.1.1.2.39 TRichView.RVData

This read-only property points to an object containing RichView document.

Each TRichView control owns one such object.

Similar objects represent cells⁽⁸⁹⁵⁾ in tables⁽⁸⁴⁶⁾.

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⁽⁹⁵⁷⁾, 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⁽¹³⁶⁾.

6.1.1.2.40 TRichView.RVFOptions

Options for loading and saving RVF (RichView Format⁽¹²⁴⁾). 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
<i>rvfoSavePicturesBody</i>	<ul style="list-style-type: none"> ▪ If set, full information about all pictures in RichView are saved in RVF. ▪ If not set, OnRVFPictureNeeded⁽³⁹³⁾ event occurs on loading.
<i>rvfoSaveControlsBody</i>	<ul style="list-style-type: none"> ▪ If set, full information about all controls in RichView are saved in RVF. ▪ If not set, OnRVFControlNeeded⁽³⁹²⁾ event occurs loading.
<i>rvfoSaveBinary</i>	<ul style="list-style-type: none"> ▪ 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.
<i>rvfoSaveBack</i>	<p style="text-align: right;"><i>(since version 1.2)</i></p> <p>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 <i>rvfoLoadBack</i> below)</p>
<i>rvfoUseStyleNames</i> <i>(almost obsolete)</i>	<p>If set, TRichView saves quoted names of text and paragraph styles instead of their indices</p> <p>Pluses:</p> <ul style="list-style-type: none"> ▪ content of file remains synchronized with collections of styles even if you delete or insert new styles; <p>Minuses:</p> <ul style="list-style-type: none"> ▪ increased file size; ▪ names of styles must be unique; ▪ files become sensitive to changing style names. <p>It's not recommended to use this options together with <i>rvfoSaveTextStyles</i> and <i>rvfoSaveParaStyles</i>.</p>
<i>rvfoSaveTextStyles</i>	<p>If set, collection of text styles associated with this RichView (TextStyles⁽⁶⁵⁴⁾ property of the linked⁽²⁵³⁾ TRVStyle component) is saved in RVF.</p> <p>See RVFTextStylesReadMode⁽²⁵¹⁾ property.</p> <p>This option affects saving style templates.*</p>
<i>rvfoSaveParaStyles</i>	<p>If set, collections of paragraph and list styles associated with this RichView (ParaStyles⁽⁶⁴⁸⁾ and ListStyles⁽⁶⁴⁶⁾ properties of the linked⁽²⁵³⁾ TRVStyle component) are saved in RVF.</p> <p>See RVFParaStylesReadMode⁽²⁵¹⁾ property.</p>

Option	Meaning
	This option affects saving style templates.*
<i>rvfoSaveLayout</i>	If set, the following properties are stored in RVF: Left ⁽²³⁸⁾ , Right ⁽²⁴⁴⁾ , Top ⁽²⁵⁵⁾ , BottomMargin ⁽²²⁵⁾ , MinTextWidth ⁽²⁴⁰⁾ , MaxTextWidth ⁽²³⁹⁾ , BiDiMode ⁽²²⁴⁾ . Layout is saved only when saving the whole document (not selection). (see <i>rvfoLoadLayout</i> below)
<i>rvfoSaveDocProperties</i>	If set, DocProperties ⁽²³¹⁾ and DocParameters ⁽²³⁰⁾ are saved in RVF. (see <i>rvfoLoadDocProperties</i> below)
<i>rvfoSaveStyleTemplatesOnlyNames</i> **	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 ⁽⁶⁹²⁾ them itself elsewhere.
<i>rvfoSaveDocObjects</i>	If set, DocObjects ⁽²²⁹⁾ are saved in RVF. (see <i>rvfoLoadDocObjects</i> below)

* If the both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included, and UseStyleTemplates⁽²⁵⁶⁾ = True, Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾ are saved as well.

** This option also used by TRichViewXML component when saving XML files.

Options for Loading

Option	Meaning
<i>rvfoLoadBack</i>	If not set, TRichView ignores background information, saved in RVF. (see <i>rvfoSaveBack</i> above)
<i>rvfoIgnoreUnknownPicFmt</i> **	<ul style="list-style-type: none"> ▪ 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
<i>rvfoIgnoreUnknownCtrls</i> **	The same as <i>rvfoIgnoreUnknownPicFmt</i> , but for controls
<i>rvfoIgnoreUnknownCtrlProperties</i>	If set, TRichView ignores errors occurred while reading inserted controls
<i>rvfoConvLargeImageIdxToZero</i> **	<ul style="list-style-type: none"> ▪ If set, too large (\geq ImageList.Count) image indices for <i>bullets</i> and <i>hotspots</i> are converted to 0.

Option	Meaning
	<ul style="list-style-type: none"> ▪ if not set, RVF reading methods return False (failure value) in such case.
<i>rvfoConvUnknownStylesToZero**</i>	<ul style="list-style-type: none"> ▪ If set, TRichView converts too large (\geq RichView.Style.TextStyles⁽⁶⁵⁴⁾.Count, or RichView.Style.ParaStyles⁽⁶⁴⁸⁾.Count) style indices to 0. ▪ If not set, RVF reading methods return False (failure value) in such case.
<i>rvfoLoadLayout</i>	If not set, TRichView ignores layout information saved in RVF (see <i>rvfoSaveLayout</i> above)
<i>rvfoLoadDocProperties</i>	If set, DocProperties ⁽²³¹⁾ and DocParameters ⁽²³⁰⁾ are loaded from RVF (but not by methods for RVF insertion) (see <i>rvfoSaveDocProperties</i> above)
<i>rvfoCanChangeUnits</i>	If set, the component can change Style ⁽⁶³⁰⁾ .Units ⁽⁶⁵⁵⁾ to the value read from RVF. This change is possible only when RVF is loaded, not inserted.
<i>rvfoLoadDocObjects</i>	If set, DocObjects ⁽²²⁹⁾ are loaded from RVF. (see <i>rvfoSaveDocObjects</i> above)
<i>rvfoIgnoreGraphicClasses**</i>	<p>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⁽⁹⁷²⁾)</p> <p>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.</p> <p>The procedure described above is used for graphic items (pictures⁽¹⁶³⁾ and hot pictures⁽¹⁶⁶⁾), background images of tables⁽⁸⁴⁹⁾ and table cells⁽⁸⁹⁶⁾. For pictures of list markers⁽⁷⁵⁹⁾, the specified graphic class is always used.</p>

** This option also used by TRichViewXML component when loading XML files.

See RVF Overview⁽¹²⁴⁾ 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)⁽²¹⁸⁾:

[*rvfoSavePicturesBody*, *rvfoSaveControlsBody*, *rvfoSaveBinary*, *rvfoSaveDocProperties*, *rvfoLoadDocProperties*, *rvfoSaveTextStyles*, *rvfoSaveParaStyles*, *rvfoSaveDocObjects*, *rvfoLoadDocObjects*];

See also:

- RVFWarnings⁽²⁵¹⁾

6.1.1.2.41 TRichView.RVFPaStylesReadMode

Specifies how paragraph styles⁽⁶⁴⁸⁾ and list styles⁽⁶⁴⁶⁾ are loaded from RVF files or streams.

property RVFPaStylesReadMode: TRVFReaderStyleMode⁽¹⁰⁰⁹⁾;

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode⁽¹⁰⁰⁹⁾ type for details.

Paragraph styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFTextStylesReadMode⁽²⁵¹⁾;
- RVFOptions⁽²⁴⁷⁾.

See also:

- RVF Overview⁽¹²⁴⁾

6.1.1.2.42 TRichView.RVFTextStylesReadMode

Specifies how text styles⁽⁶⁵⁴⁾ are loaded from RVF files or streams.

property RVFTextStylesReadMode: TRVFReaderStyleMode⁽¹⁰⁰⁹⁾;

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode⁽¹⁰⁰⁹⁾ type for details.

Text styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFPaStylesReadMode⁽²⁵¹⁾;
- RVFOptions⁽²⁴⁷⁾.

See also:

- RVF Overview⁽¹²⁴⁾.

6.1.1.2.43 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⁽²⁴⁷⁾ some cases could be considered as warnings (and reading method was successfully finished) or errors.

Warning	Meaning
<i>rvfwUnknownPicFmt</i>	Pictures of unknown classes exist in RVF. See <i>rvfoIgnoreUnknownPicFmt</i> .
<i>rvfwUnknownCtrls</i>	Controls of unknown classes exist in RVF. See <i>rvfoIgnoreUnknownCtrls</i> .
<i>rvfwConvLargeImageIdx</i>	Bullets or hotspots with too large image indices exist in RVF. See <i>rvfoConvLargeImageIdxToZero</i> .
<i>rvfwConvUnknownStyles</i>	Text, paragraph or list styles with too large indices exist in RVF. See <i>rvfoConvUnknownStylesToZero</i> .
<i>rvfwConvToUnicode</i> <i>rvfwConvFromUnicode</i>	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.
<i>rvfwInvalidPicture</i>	Some pictures were replaced with InvalidPicture ⁽⁶⁴³⁾ .
<i>rvfwUnknownCtrlProperties</i>	There was error when reading inserted controls (for example, unknown properties)
<i>rvfwConvUnits</i>	RVF file has different units than Style ⁽⁶³⁰⁾ .Units ⁽⁶⁵⁵⁾ , and a conversion of units was made. This warning is not set if units were changed because of <i>rvfoCanChangeUnits</i> included in RVFOptions ⁽²⁴⁷⁾ . A conversion is performed automatically, without generating errors.

See RVF Overview⁽¹²⁴⁾ for details.

Methods reading RVF:

- LoadRVFFromStream⁽³²⁹⁾;
- InsertRVFFromStream⁽³¹⁶⁾;
- AppendRVFFromStream⁽²⁷⁶⁾;
- LoadRVF⁽³²⁸⁾.

Methods of RichViewEdit reading RVF:

- `InsertRVFFromFileEd`⁵²⁸;

- `InsertRVFFromStreamEd`⁵²⁹;

(also `PasteRVF`⁵³⁸ and `Paste`⁵³⁴, but in this version you can't detect when Clipboard reading operation is finished).

6.1.1.2.44 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`³⁹⁶, if `ssTouch` is included in the `Shift` parameter of this event.

Default value:

True

See also properties of TRVStyle:

- `SelectionHandleKind`⁶⁴⁸

6.1.1.2.45 TRichView.SingleClick

property `SingleClick` : `Boolean`;

Obsolete, please use `rvoSingleClick` in `Options`²⁴⁰;

6.1.1.2.46 TRichView.Style

Style of TRichView: 

property `Style`: `TRVStyle`⁶³⁰;

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.

6.1.1.2.47 TRichView.StyleTemplateInsertMode

Controls how Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾ 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⁽²⁵⁶⁾=True.

Mode	Meaning
<i>rvstimUseTargetFormatting</i>	Use formatting of the current (target) document
<i>rvstimUseSourceFormatting</i>	Use formatting of the inserted (source) document
<i>rvstimIgnoreSourceStyleTemplates</i>	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⁽⁶⁸⁹⁾ (if it exists). The inserted text is unstyled, except for hyperlinks⁽⁷¹⁴⁾ (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`⁽¹⁰⁴³⁾ = `False`. In the mode (3), "Normal" is always applied.

Default value

`rvstimUseTargetFormatting`

See also:

- Styles and style templates⁽⁹⁸⁾.

6.1.1.2.48 TRichView.TabNavigation

Mode of **Tab** and **Shift + Tab** navigation

type

```
TRVTabNavigationType =
  (rvtnNone, rvtnTab, rvtnCtrlTab)
```

property `TabNavigation`: `TRVTabNavigationType`;

(introduced in version 1.3)

Mode	Meaning
<code>rvtnNone</code>	No special actions for Tab (default for <code>TRichViewEdit</code>). A focus (dotted) rectangle is not drawn around hyperlinks, it's impossible to open hyperlink with Enter .
<code>rvtnTab</code>	Tab-navigation through links and controls using Tab and Shift + Tab (default for <code>TRichView</code>);
<code>rvtnCtrlTab</code>	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`⁽²⁹⁵⁾ method.

6.1.1.2.49 TRichView.TopMargin

Top margin of document

property `TopMargin`: `TRVPixelLength`⁽¹⁰¹⁹⁾;

This value is measured in "standard" pixels, 1 "standard" pixel = $1/\text{Style}^{(253)} \cdot \text{UnitsPerInch}^{(655)}$ of an inch. When drawing on a device (screen or printer), this value is recalculated according to the device DPI (dots per inch) value.

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetIntPropertyEd`⁽⁵⁴⁵⁾.

See also properties:

- LeftMargin⁽²³⁸⁾;
- RightMargin⁽²⁴⁴⁾;
- BottomMargin⁽²²⁵⁾.

See also properties of TRVPrint:

- Margins⁽⁷⁶⁸⁾.

See also:

Additional information about horizontal scrolling, margins, indents, etc.⁽¹⁰⁵⁾

6.1.1.2.50 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⁽²⁵⁷⁾
- DarkMode⁽²²⁸⁾

6.1.1.2.51 TRichView.UseStyleTemplates

Allows or disallows using Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾.

property UseStyleTemplates: Boolean;

(introduced in version 14)

If UseStyleTemplates=*True*:

- StyleTemplates are saved in RVF (also requires both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* included in RVFOptions⁽²⁴⁷⁾);
- StyleTemplates are loaded from RVF (also requires RVFTextStylesReadMode⁽²⁵¹⁾ =RVFParaStylesReadMode⁽²⁵¹⁾=*rvf_sInsertMerge*);
- StyleTemplates are saved in RTF and DocX (as a style sheet);
- StyleTemplates are loaded from RTF and DocX (also requires RTFReadProperties⁽²⁴⁶⁾.TextStyleMode⁽⁴⁵⁸⁾=RTFReadProperties.ParaStyleMode⁽⁴⁵⁶⁾=*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⁽²⁵⁴⁾;
- "Normal" style template (if it exists) is assigned to unstyled paragraphs when loading RTF and RVF files/streams (see RichViewAutoAssignNormalStyleTemplate⁽¹⁰⁴³⁾);

- the component provides that `TextStyle.ParaStyleTemplateId`⁷⁰⁵ = `ParaStyle.StyleTemplateId`⁶⁹⁶ for all text items¹⁶¹ and non-text items linked to a text style (tabs¹⁶², labels¹⁷⁶, sequences¹⁷⁷, footnotes¹⁸⁰, endnotes¹⁷⁸, note references¹⁸⁴); if they are not equal, the component assigns another text style to this item, with the correct `ParaStyleTemplateId`, adding this text style to `Style`⁶³⁰.`TextStyles`⁶⁵⁴ if necessary.
- in editor, when **Enter** is pressed, the next paragraph is formatted according to `NextId`⁷³⁶ property of the current paragraph's style template

Default value

False

See also:

- Styles and style templates⁹⁸.

6.1.1.2.52 TRichView.UseVCLThemes

Allows using colors of VCL or custom themes.

```
property UseVCLThemes: Boolean;
```

(introduced in version 17)

If *True*:

- if `RVGetSystemColor`¹⁰⁵⁷ 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`²²⁸
- `UseFMXThemes`²⁵⁶

6.1.1.2.53 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`²⁸⁰ method.

6.1.1.2.54 TRichView.VSmallStep

Number of pixels in one step of vertical scrollbar

```
property VSmallStep: Integer;
```

When `VScrollPos`⁸¹⁹ is changes by 1, the window is scrolled by `VSmallStep` pixels.

This is one "RVSU", see `Scrolling in RichView`¹⁰⁵

Default value:

10

See also properties:

- VScrollPos ⁸¹⁹.
- VScrollMax ⁸¹⁸.
- VScrollVisible ⁸¹⁹.

6.1.1.2.55 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 ⁷²³ of paragraph style, this property should be considered as a control (view) property, not as a document property. It is not stored in RVF ¹²⁴.

Default value:*True*

6.1.1.2.56 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* ²⁸⁸ or *Reformat* ³³².

Zooming is applied to almost everything: margins, background images, items in a document. However, there are exceptions; the following items are not affected:

- controls ¹⁶⁸.

You need to scale controls yourself.

Default value:

100 (meaning 100% zoom)

See also:

- *DocumentPixelsPerInch* ²³³ (analog for VCL)

6.1.1.3 Methods

Derived from TCustomRichView

*AddBreak* ²⁶²*AddBullet* ²⁶³

-  [AddCheckpoint](#) ⁽²⁶⁴⁾
-  [AddControl](#) ⁽²⁶⁵⁾
-  [AddFmt](#) ⁽²⁶⁶⁾
-  [AddHotPicture](#) ⁽²⁶⁷⁾
-  [AddHotspot](#) ⁽²⁶⁸⁾
-  [AddItem](#) ⁽²⁶⁹⁾
-  [AddNL -A -W, Add](#) ⁽²⁷⁰⁾
-  [AddPicture](#) ⁽²⁷²⁾
-  [AddTab](#) ⁽²⁷³⁾
-  [AddTextNL -A -W](#) ⁽²⁷⁴⁾
-  [AppendFrom](#) ⁽²⁷⁵⁾
-  [AppendRVFFromStream](#) ⁽²⁷⁶⁾
- [AssignSoftPageBreaks](#) ⁽²⁷⁷⁾
- [BeginOleDrag](#) ⁽²⁷⁸⁾ [VCL, LCL]
-  [Clear](#) ⁽²⁷⁹⁾
- [ClearLiveSpellingResults](#) ⁽²⁷⁹⁾
- [ClearSoftPageBreaks](#) ⁽²⁷⁹⁾
- [ClientToDocument, DocumentToClient](#) ⁽²⁸⁰⁾
- [ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits](#) ⁽²⁸⁰⁾
- [Copy](#) ⁽²⁸¹⁾
- [CopyDef](#) ⁽²⁸¹⁾
- [CopyImage](#) ⁽²⁸¹⁾
- [CopyRTF](#) ⁽²⁸²⁾
- [CopyRVF](#) ⁽²⁸²⁾
- [CopyText -A -W](#) ⁽²⁸³⁾
- [Create](#) ⁽²⁸³⁾
-  [DeleteItems](#) ⁽²⁸³⁾
- [DeleteMarkedStyles](#) ⁽²⁸⁴⁾
- [DeleteParas](#) ⁽²⁸⁵⁾
-  [DeleteSection](#) ⁽²⁸⁵⁾
- [DeleteUnusedStyles](#) ⁽²⁸⁶⁾
- [Deselect](#) ⁽²⁸⁶⁾
- [Destroy](#) ⁽²⁸⁶⁾
- [FindCheckPointByName](#) ⁽²⁸⁷⁾
- [FindCheckPointByTag](#) ⁽²⁸⁷⁾
- [FindControlItemNo](#) ⁽²⁸⁷⁾
- [Format](#) ⁽²⁸⁸⁾
- [FormatTail](#) ⁽²⁸⁸⁾
- [GetBreakInfo](#) ⁽²⁸⁹⁾
- [GetBulletInfo](#) ⁽²⁸⁹⁾
- [GetCheckpointByNo](#) ⁽²⁹¹⁾
- [GetCheckpointInfo](#) ⁽²⁹¹⁾
- [GetCheckpointItemNo](#) ⁽²⁹¹⁾
- [GetCheckpointNo](#) ⁽²⁹²⁾
- [GetCheckpointXY](#) ⁽²⁹²⁾
- [GetCheckpointY](#) ⁽²⁹²⁾
- [GetCheckpointYEx](#) ⁽²⁹³⁾

- GetControlInfo²⁹³
- GetFirstCheckpoint²⁹⁴
- GetFocusedItem²⁹⁵
- GetHotspotInfo²⁹⁵
- GetItem²⁹⁶
- GetItemAt²⁹⁷
- GetItemCheckpoint²⁹⁸
- GetItemCoords²⁹⁸
- GetItemCoordsEx²⁹⁹
- GetItemExtraIntProperty³⁰⁰
- GetItemExtraIntPropertyEx³⁰⁰
- GetItemExtraStrProperty³⁰⁰
- GetItemExtraStrPropertyEx³⁰⁰
- GetItemNo³⁰¹
- GetItemPara³⁰¹
- GetItemStyle³⁰²
- GetItemTag³⁰²
- GetItemText -A -W³⁰³
- GetItemVAlign³⁰³
- GetJumpPointLocation³⁰⁴
- GetJumpPointY³⁰⁵
- GetLastCheckpoint³⁰⁵
- GetLineNo³⁰⁶
- GetListMarkerInfo³⁰⁷
- GetNextCheckpoint³⁰⁷
- GetOffsAfterItem³⁰⁸
- GetOffsBeforeItem³⁰⁹
- GetPictureInfo³¹⁰
- GetPrevCheckpoint³¹¹
- GetRealDocumentPixelsPerInch³¹¹
- GetSelectedImage³¹²
- GetSelectionBounds³¹²
- GetSelText -W³¹³
- GetTextInfo³¹³
- GetWordAt -A -W³¹⁴
-  InsertRVFFromStream³¹⁶
- IsFromNewLine³¹⁷
- IsParaStart³¹⁸
- LiveSpellingValidateWord³¹⁸
-  LoadDocX³¹⁸ (D2009+ or with Delphi ZLib)
-  LoadDocXFromStream³¹⁹ (D2009+ or with Delphi ZLib)
-  LoadFromFile³²⁰
-  LoadFromFileEx³²⁰
-  LoadFromStream³²¹
-  LoadFromStreamEx³²¹
-  LoadHTML³²²
-  LoadHTMLFromStream³²⁴

-  LoadMarkdown ³²⁵
-  LoadMarkdownFromStream ³²⁵
-  LoadRTF ³²⁶
-  LoadRTFFromStream ³²⁷
-  LoadRVF ³²⁸
-  LoadRVFFromStream ³²⁹
-  LoadText -W ³³⁰
-  LoadTextFromStream -W ³³¹
- MarkStylesInUse ³³¹
- RefreshListMarkers ³³²
- Reformat ³³²
-  RemoveCheckpoint ³³²
- ResetAnimation ³³³
- SaveDocX ³³³
- SaveDocXToStream ³³⁴
- SaveHTML ³³⁵
- SaveHTMLToStream ³³⁸
- SavePicture ³⁴²
- SaveMarkdown ³⁴⁰
- SaveMarkdownToStream ³⁴¹
- SaveRTF ³⁴³
- SaveRTFToStream ³⁴⁴
- SaveRVF ³⁴⁴
- SaveRVFToStream ³⁴⁴
- SaveText -W ³⁴⁵
- SaveTextToStream -W ³⁴⁵
- SearchText -A -W ³⁴⁶
- SelectAll ³⁴⁸
- SelectControl ³⁴⁸
- SelectionExists ³⁴⁹
- SelectWordAt ³⁴⁹
- SetAddParagraphMode ³⁵⁰
-  SetBreakInfo ³⁵¹
-  SetBulletInfo ³⁵²
-  SetCheckpointInfo ³⁵³
-  SetControlInfo ³⁵⁴
- SetFooter ³⁵⁵
- SetHeader ³⁵⁶
-  SetHotspotInfo ³⁵⁷
-  SetItemExtraIntProperty ³⁵⁸
-  SetItemExtraIntPropertyEx ³⁵⁸
-  SetItemExtraStrProperty ³⁵⁹
-  SetItemExtraStrPropertyEx ³⁵⁹
-  SetItemTag ³⁶⁰
-  SetItemText -A -W ³⁶⁰
-  SetItemVAlign ³⁶¹

-  [SetListMarkerInfo](#) ⁽³⁶²⁾
-  [SetPictureInfo](#) ⁽³⁶³⁾
- [SetSelectionBounds](#) ⁽³⁶⁵⁾
- [StartAnimation](#) ⁽³⁶⁶⁾
- [StartLiveSpelling](#) ⁽³⁶⁷⁾
- [StopAnimation](#) ⁽³⁶⁷⁾
- [StoreSearchResult](#) ⁽³⁶⁷⁾
- [UpdatePaletteInfo](#) ⁽³⁶⁸⁾

Derived from TRVScroller ⁽⁸¹³⁾

[ScrollTo](#) ⁽⁸²⁰⁾

6.1.1.3.1 TRichView.AddBreak

Adds a *break* ⁽¹⁶⁷⁾ (a horizontal line) to the end of document.

```
procedure AddBreak(Width: TRVStyleLength (1027) = 0;
  Style: TRVBreakStyle (995) = rvbsLine; Color: TRVColor (996) = rvclNone (1038);
const Tag: TRVTag (1029) = RVEMPTYTAG (1056));
```

(changed in version 18)

Parameters:

Width – line width (or rectangle height). This value is measured in `Style` ⁽²⁵³⁾.Units ⁽⁶⁵⁵⁾. If **Width** = 0, the new item has 1 pixel width (converted to `Style` ⁽²⁵³⁾.Units ⁽⁶⁵⁵⁾)

Style – break type;

Color – color of line. If it is equal to `rvclNone` ⁽¹⁰³⁸⁾, the *break* color is `Style` ⁽²⁵³⁾.TextStyles ⁽⁶⁵⁴⁾ [0].Color ⁽⁷⁰¹⁾.

Tag – `tag` ⁽⁹¹⁾ of this *break*.

For default HTML `<hr>` appearance, parameters must be: **Style**=`rvbs3d`, **Width**=2.

Methods type:  viewer-style.

See also methods:

- [Format](#) ⁽²⁸⁸⁾;
- [FormatTail](#) ⁽²⁸⁸⁾.

See also methods of TRichViewEdit:

- [InsertBreak](#) ⁽⁵¹⁵⁾.

See also:

- [Item types](#) ⁽⁸⁵⁾;
- [Other methods for appending items](#) ⁽¹⁰²⁾;
- ["Tags"](#) ⁽⁹¹⁾.

Related deprecated methods:

```
procedure AddBreakExTag(Width: TRVStyleLength (1027); Style: TRVBreakStyle (995);
  Color: TRVColor (996); const Tag: TRVTag (1029));
procedure AddBreakEx(Width: TRVStyleLength (1027); Style: TRVBreakStyle (995);
  Color: TRVColor (996));
```

```
procedure AddBreakTag(const Tag: TRVTag1029);
```

These methods provide subset of functionality of AddBreak.

6.1.1.3.2 TRichView.AddBullet

Adds a *bullet*¹⁷⁰ (an image from TImageList) to the end of document.

VCL and LCL:

```
procedure AddBullet(const AName: TRVUnicodeString1032;
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign1033 = rvvaBaseline;
  const Tag: TRVTag1029 = RVEMPTYTAG1056);
```

(changed in version 18)

FireMonkey:

```
procedure AddBullet(const AName: TRVUnicodeString1032;
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign1033 = rvvaBaseline;
  const Tag: TRVTag1029 = RVEMPTYTAG1056;
  ImageWidth: TRVStyleLength1027 = 16;
  ImageHeight: TRVStyleLength1027 = 16);
```

Alternatives:

- If you want to add a hyperlink-picture from image list, add a *hotspot*²⁶⁸ instead.
- If you want to add a normal picture, add a *picture*²⁷² instead.
- If you want to add a normal picture-hyperlink, add a *hot picture*²⁶⁷ 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**²⁵³.**ParaStyles**⁶⁴⁸[**ParaNo**].

Tag – *tag*⁹¹ 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**²⁸⁸;
- **FormatTail**²⁸⁸;
- **SetAddParagraphMode**³⁵⁰.

See also methods of TRichViewEdit:

- InsertBullet⁽⁵¹⁵⁾.

See also properties of TRVStyle:

- ParaStyles⁽⁶⁴⁸⁾.

See also:

- Item types⁽⁸⁵⁾;
- Other methods for appending items⁽¹⁰²⁾;
- "Tags"⁽⁹¹⁾.

Related deprecated methods:

```

procedure AddBulletExTag(const Name: TRVUnicodeString(1032);
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer; const Tag: TRVTag(1029));
procedure AddBulletEx(const Name: TRVUnicodeString(1032);
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer);

```

These methods provide subset of functionality of AddBullet.

6.1.1.3.3 TRichView.AddCheckpoint

Adds a *checkpoint*⁽⁸⁷⁾ to the end of document.

```

function AddCheckpoint(const CpName: TRVUnicodeString(1032) = '';
  RaiseEvent: Boolean = False;
  const CpTag: TRVTag(1029) = RVEMPTYTAG(1056)): 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⁽³⁷²⁾

Tag – *tag* of *checkpoint*.

Return value:

Index of the added checkpoint

Methods type:  viewer-style.

See also methods:

- SetCheckpointInfo⁽³⁵³⁾;
- RemoveCheckpoint⁽³³²⁾.

See also methods of TRichViewEdit:

- SetCheckpointInfoEd⁽⁵⁴⁹⁾;
- SetCurrentCheckpointInfo⁽⁵⁵³⁾.

See also:

- Methods for appending items⁽¹⁰²⁾;

- *Checkpoints*⁽⁸⁷⁾;
- *Tags*⁽⁹¹⁾.

Related deprecated methods:

```
function AddNamedCheckpointExTag(const CpName: TRVUnicodeString(1032);
  RaiseEvent: Boolean; const Tag: TRVTag(1029)): Integer;
function AddCheckpoint: Integer;
function AddCheckpointTag(const Tag: TRVTag(1029)): Integer;
function AddNamedCheckpoint(const CpName: TRVUnicodeString(1032)): Integer;
function AddNamedCheckpointEx(const CpName: TRVUnicodeString(1032);
  RaiseEvent: Boolean): Integer;
```

These methods provide subset of functionality of AddCheckpoint.

6.1.1.3.4 TRichView.AddControl

These methods add Delphi control⁽¹⁶⁸⁾ to the end of document.

```
procedure AddControl(const Name: TRVUnicodeString(1032); ctrl: TControl;
  ParaNo: Integer = -1; VAlign: TRVVAlign(1033) = rvvaBaseline;
  const Tag: TRVTag(1029) = RVEMPTYTAG(1056));
```

(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⁽¹⁰³³⁾ 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⁽²⁵³⁾.ParaStyles⁽⁶⁴⁸⁾[**ParaNo**].

Tag – tag⁽⁹¹⁾ of this control item. Do not confuse with **ctrl.Tag** property.

Methods type:  viewer-style.

See also methods:

- Format⁽²⁸⁸⁾;
- FormatTail⁽²⁸⁸⁾;
- SetAddParagraphMode⁽³⁵⁰⁾.

See also methods of TRichViewEdit:

- InsertControl⁽⁵¹⁷⁾.

See also properties of TRVStyle:

- ParaStyles⁽⁶⁴⁸⁾.

See also:

- Item types⁽⁸⁵⁾;
- Other methods for appending items⁽¹⁰²⁾;
- "Tags"⁽⁹¹⁾.

Related deprecated methods:

```

procedure AddControlExTag(const Name: TRVAnsiString993; ctrl: TControl;
  ParaNo: Integer; VAlign: TRVVAlign1033; const Tag: TRVTag1029);
procedure AddControlEx(const Name: TRVAnsiString993; ctrl: TControl;
  ParaNo: Integer; VAlign: TRVVAlign1033);

```

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 AddNL²⁷⁰(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 documents¹³⁸.

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 AddTextNL²⁷⁴.

Args – arguments for format string.

StyleNo is an index in the TextStyles⁶⁵⁴ collection of the linked²⁵³ RVStyle component, or rvsDefStyle¹⁰⁵⁹ 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 Style²⁵³.ParaStyles⁶⁴⁸[**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 Lazarus¹⁵⁴

Example:

```
MyRichView.AddFmt('X=%d and Y=%d', [X,Y], 0, 0);
```

If style templates²⁵⁶ 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:

- AddNL²⁷⁰;
- AddTextNL²⁷⁴;
- Format²⁸⁸;

- `FormatTail`²⁸⁸;
- `SetAddParagraphMode`³⁵⁰.

See also properties of TRVStyle:

- `TextStyles`⁶⁵⁴;
- `ParaStyles`⁶⁴⁸;
- `SpacesInTab`⁶⁵².

See also:

- Other methods for appending items¹⁰².

6.1.1.3.6 TRichView.AddHotPicture

Adds picture-hyperlink¹⁶⁶ to the end of document.

```
procedure AddHotPicture(const Name: TRVUnicodeString1032;
  gr: TRVGraphic970; ParaNo: Integer = -1;
  VAlign: TRVVAlign1033 = rvvaBaseline;
  const Tag: TRVTag1029 = RVEMPTYTAG1056;
  ImageWidth: TRVStyleLength1027 = 0;
  ImageHeight: TRVStyleLength1027 = 0);
```

(introduced in version 1.5; changed in versions 18 and 21)

The same as `AddPicture`²⁷², 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 property¹⁰⁰⁰, 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`¹⁰⁴⁶). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see `TRVVAlign`¹⁰³³ 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`²⁵³.`ParaStyles`⁶⁴⁸[**ParaNo**].

Tag – `tag`⁹¹ 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`¹⁰⁰⁰).

Alternatives:

- If you want to add a simple (not a hyperlink) picture, add a picture²⁷² instead.
- If you want to add a hyperlink-picture from image list, add a *hotspot*²⁶⁸ instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet*²⁶³ instead.

Methods type:  viewer-style.

See also methods:

- `Format`²⁸⁸;
- `FormatTail`²⁸⁸;

- `SetAddParagraphMode` ⁽³⁵⁰⁾.

See also methods of `TRichViewEdit`:

- `InsertHotPicture` ⁽⁵¹⁹⁾.

See also properties of `TRVStyle`:

- `ParaStyles` ⁽⁶⁴⁸⁾.

See also:

- Item types ⁽⁸⁵⁾;
- Other methods for appending items ⁽¹⁰²⁾;
- "Tags" ⁽⁹¹⁾;
- Hypertext in `TRichView` ⁽⁹²⁾.

Related deprecated methods:

```
procedure AddHotPictureTag(const Name: TRVUnicodeString1032;
  gr: TGraphic; ParaNo: Integer;
  VAlign: TRVVAlign1033; const Tag: TRVTag1029);
```

6.1.1.3.7 TRichView.AddHotspot

Adds a *hotspot* ⁽¹⁷²⁾ (an image from `TImageList` – hypertext link) to the end of document.

VCL and LCL:

```
procedure AddHotspot(const AName: TRVUnicodeString1032;
  ImageIndex, HotImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign1033 = rvvaBaseline;
  const Tag: TRVTag1029 = RVEMPTYTAG1056);
```

(changed in version 18)

FireMonkey:

```
procedure AddHotspot(const AName: TRVUnicodeString1032;
  ImageIndex, HotImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign1033 = rvvaBaseline;
  const Tag: TRVTag1029 = RVEMPTYTAG1056;
  ImageWidth: TRVStyleLength1027 = 16;
  ImageHeight: TRVStyleLength1027 = 16);
```

Alternatives:

- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet* ⁽²⁶³⁾ instead.
- If you want to add a picture-hyperlink, add a *hot picture* ⁽²⁶⁷⁾ instead.
- If you want to add a picture, add a *picture* ⁽²⁷²⁾ 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**⁽²⁵³⁾.**ParaStyles**⁽⁶⁴⁸⁾ [**ParaNo**].

Tag – *tag*⁽⁹¹⁾ 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**⁽²⁸⁸⁾;
- **FormatTail**⁽²⁸⁸⁾;
- **SetAddParagraphMode**⁽³⁵⁰⁾.

See also methods of **TRichViewEdit**:

- **InsertHotspot**⁽⁵²⁰⁾.

See also properties of **TRVStyle**:

- **ParaStyles**⁽⁶⁴⁸⁾.

See also:

- Item types⁽⁸⁵⁾;
- Other methods for appending items⁽¹⁰²⁾;
- "Tags"⁽⁹¹⁾;
- Hypertext in **RichView**⁽⁹²⁾.

Related deprecated methods:

```

procedure AddHotspotExTag(const Name: TRVUnicodeString(1032);
  ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer;
  const Tag: TRVTag(1029));
procedure AddHotspotEx(const Name: TRVUnicodeString(1032);
  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(1032); Item: TCustomRVItemInfo(844));
(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 ⁽¹⁰²⁾.

Usually this method is used for appending tables ⁽¹⁷³⁾, labels ⁽¹⁷⁶⁾, sequences ⁽¹⁷⁷⁾, footnotes ⁽¹⁸⁰⁾, endnotes ⁽¹⁷⁸⁾, references to notes ⁽¹⁸⁴⁾.

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 ⁽⁸⁴⁴⁾.

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 ⁽⁶⁴⁸⁾)

Method type:  viewer-style.

See also methods:

- Format ⁽²⁸⁸⁾;
- FormatTail ⁽²⁸⁸⁾;
- SetAddParagraphMode ⁽³⁵⁰⁾;
- GetItem ⁽²⁹⁶⁾.

See also methods of TRichViewEdit:

- InsertItem ⁽⁵²²⁾.

See also:

- Item types ⁽⁸⁵⁾;
- Other methods for appending items ⁽¹⁰²⁾.

6.1.1.3.9 TRichView.AddNL -A -W

These methods add one text item ⁽¹⁶¹⁾ 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 (1029) = RVEEMPTYTAG (1056));
procedure AddNLW(const s: TRVUnicodeString (1032); StyleNo, ParaNo: Integer;
  const Tag: TRVTag (1029) = RVEEMPTYTAG (1056));
procedure AddNLA(const s: TRVAnsiString (993); StyleNo, ParaNo: Integer;
  const Tag: TRVTag (1029) = RVEEMPTYTAG (1056));
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 ⁽²⁷⁴⁾.

StyleNo is an index in the TextStyles ⁽⁶⁵⁴⁾ collection of the linked ⁽²⁵³⁾ RVStyle component, or *rvsDefStyle* ⁽¹⁰⁵⁹⁾ 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(253).ParaStyles(648)[ParaNo]`.

Tag – `tag(91)` 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(138)`.

Unicode notes:

Internally, text is stored as Unicode. **AddNLA** converts ANSI to Unicode. A code page for conversion is calculated basing on `Style(253).TextStyles(654)[StyleNo]`. `Charset(700)`. If it is equal to `DEFAULT_CHARSET`, and for non-text items, `Style(253).DefCodePage(635)` is used.

AddNL works:

- like **AddNLA**, in Delphi 2007 and older
- like **AddNLW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus⁽¹⁵⁴⁾

Methods type:  viewer-style

If style templates⁽²⁵⁶⁾ 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(273)`;
- `Format(288)`;
- `FormatTail(288)`;
- `SetAddParagraphMode(350)`.

See also methods of `TRichViewEdit`:

- `InsertStringTag`, `InsertStringATag`, `InsertStringWTag(530)`.

See also properties of `TRVStyle`:

- `TextStyles(654)`;
- `SpacesInTab(652)`.

See also:

- Other methods for appending items⁽¹⁰²⁾;
- Unicode in `RichView(130)`.

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⁽¹⁶³⁾ to the end of document

```
procedure AddPicture(const Name: TRVUnicodeString(1032); gr: TRVGraphic(803);
  ParaNo: Integer = -1; VAlign: TRVVAlign(1033) = rvvaBaseline;
  const Tag: TRVTag(1029) = RVEMPTYTAG(1056); ImageWidth: TRVStyleLength(1027) = 0;
  ImageHeight: TRVStyleLength(1027) = 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⁽¹⁰⁰⁰⁾, 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⁽¹⁰⁴⁶⁾). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign⁽¹⁰³³⁾ 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⁽²⁵³⁾.ParaStyles⁽⁶⁴⁸⁾[**ParaNo**].

Tag – tag⁽⁹¹⁾ 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_TRVExtraltemProperty⁽¹⁰⁰⁰⁾).

Alternatives:

- If you want to add a picture-hyperlink, add a *hot picture*⁽²⁶⁷⁾ instead.
- If you want to add a hyperlink-picture from image list, add a *hotspot*⁽²⁶⁸⁾ instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a *bullet*⁽²⁶³⁾ 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(279);
AddBitmapFromFile(rv, 'd:\pic1.bmp', 0);
AddBitmapFromFile(rv, 'd:\pic2.bmp', -1);
rv.Format(288);
```

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.SetItemExtraIntProperty358(rv.ItemCount237-1, rvepShared, 1);
rv.AddPicture('', bmp, -1);
rv.SetItemExtraIntProperty358(rv.ItemCount237-1, rvepShared, 1);
rv.Format288;
// you must free bmp yourself when rv is cleared.

```

See also methods:

- Format²⁸⁸;
- FormatTail²⁸⁸;
- SetAddParagraphMode³⁵⁰.

See also methods of TRichViewEdit:

- InsertPicture⁵²⁶.

See also properties of TRVStyle:

- ParaStyles⁶⁴⁸.

See also:

- Item types⁸⁵;
- Other methods for appending items¹⁰²;
- "Tags"⁹¹.

Related deprecated methods:

```

procedure AddPictureExTag(const Name: TRVUnicodeString1032; gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign1033; const Tag: TRVTag1029);
procedure AddPictureEx(const Name: TRVUnicodeString1032; gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign1033);

```

These methods provide subset of functionality of AddPicture.

6.1.1.3.11 TRichView.AddTab

Adds tabulator¹⁶² 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 TextStyles⁶⁵⁴ collection of the linked²⁵³ RVStyle component, or *rvsDefStyle*¹⁰⁵⁹ 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 Style²⁵³.ParaStyles⁶⁴⁸ [**ParaNo**].

This method always adds a tabulator, regardless of RVStyle.SpacesInTab⁶⁵² mode. If RVStyle.SpacesInTab⁶⁵²=0, this method does the same work as AddTextNL²⁷⁴ (#9, TextStyleNo, ParaNo, ParaNo).

Methods type:  viewer-style.

See also methods:

- `Format`⁽²⁸⁸⁾;
- `FormatTail`⁽²⁸⁸⁾;
- `SetAddParagraphMode`⁽³⁵⁰⁾.

See also methods of TRichViewEdit:

- `InsertTab`⁽⁵³¹⁾.

See also properties of TRVStyle:

- `ParaStyles`⁽⁶⁴⁸⁾.

See also:

- Item types⁽⁸⁵⁾;
- Other methods for appending items⁽¹⁰²⁾.

6.1.1.3.12 TRichView.AddTextNL -A -W

These methods add one or more text⁽¹⁶¹⁾ lines to the end of document.

```

procedure AddTextNL(const s: String;
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag(1029) = RVEMPTYTAG(1056));
procedure AddTextNLA(const s: TRVAnsiString(993);
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag(1029) = RVEMPTYTAG(1056);
  CodePage: TRVCodePage = CP_ACP);
procedure AddTextNLW(const S: TRVUnicodeString(1032);
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  DefAsSingleParagraph: Boolean = False;
  const Tag: TRVTag(1029) = RVEMPTYTAG(1056));

```

(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⁽¹⁶²⁾, depending on value of `SpacesInTab`⁽⁶⁵²⁾ property of the linked⁽²⁵³⁾ RVStyle component. #12 characters add page breaks⁽²⁴⁴⁾. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported.

StyleNo is an index in the `TextStyles`⁽⁶⁵⁴⁾ collection of the linked⁽²⁵³⁾ RVStyle component, or `rvsDefStyle`⁽¹⁰⁵⁹⁾ 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`⁽²⁵³⁾.`ParaStyles`⁽⁶⁴⁸⁾[**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`⁽³⁵⁰⁾). 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`⁽³⁵⁰⁾). If *True*, it treats #13 and #10 as line breaks, not paragraph breaks.

Tag – *tag*⁽⁹¹⁾ 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`⁽²⁵³⁾.`DefCodePage`⁽⁶³⁵⁾ 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⁽¹⁵⁴⁾

If style templates⁽²⁵⁶⁾ 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⁽²⁷⁰⁾;
- `AddTab`⁽²⁷³⁾;
- `LoadText`, `LoadTextW`⁽³³⁰⁾;
- `LoadTextFromStream`, `LoadTextFromStreamW`⁽³³¹⁾;
- `Format`⁽²⁸⁸⁾;
- `FormatTail`⁽²⁸⁸⁾;
- `SetAddParagraphMode`⁽³⁵⁰⁾.

See also methods of TRichViewEdit:

- `InsertText`, `-A`, `-W`⁽⁵³²⁾.

See also properties of TRVStyle:

- `TextStyles`⁽⁶⁵⁴⁾;
- `SpacesInTab`⁽⁶⁵²⁾.

See also:

- Other methods for appending items⁽¹⁰²⁾;
- Unicode in RichView⁽¹³⁰⁾;
- Example how to load UTF-8 files⁽⁴⁶⁰⁾.

6.1.1.3.13 TRichView.AppendFrom

Adds content of another RichView component (**Source**) to the end of document.

```
procedure AppendFrom(Source: TCustomRichView(210));
```

This method does not copy items of some types (including inserted controls⁽¹⁶⁸⁾ and tables⁽¹⁷³⁾) (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(344)(Stream, False);
    Stream.Position := 0;
    Dest.LoadRVFFromStream(329)(Stream);
    Dest.Format(288);
  finally
    Stream.Free;
  end;
end;
```

Method type:  viewer-style.

See also methods:

- Format⁽²⁸⁸⁾;
- FormatTail⁽²⁸⁸⁾.

See also:

- Other methods for appending items⁽¹⁰²⁾.

6.1.1.3.14 TRichView.AppendRVFFromStream

Adds data from Stream in RichView Format (RVF)⁽¹²⁴⁾ 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⁽⁶⁴⁸⁾ collection of the linked⁽²⁵³⁾ 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⁽²¹⁸⁾.

If style templates are used⁽²⁵⁶⁾, and RVFTextStylesReadMode⁽²⁵¹⁾=RVFParaStylesReadMode⁽²⁵¹⁾=rvf_sInsertMerge, and StyleTemplateInsertMode⁽²⁵⁴⁾<>rvstimIgnoreSourceStyleTemplates, the method merges style templates from RVF into Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾, and reads text and paragraph styles according to StyleTemplateInsertMode⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange⁽⁴¹⁰⁾ event.

Return value:

"Was reading successful?"

See also properties:

- RVFOptions²⁴⁷;
- RVFWarnings²⁵¹;
- RVFTextStylesReadMode²⁵¹ (this method type: RVF insertion);
- RVFParaStylesReadMode²⁵¹ (this method type: RVF insertion);
- UseStyleTemplates²⁵⁶.

See also events:

- OnRVFImageListNeeded³⁹³;
- OnRVFControlNeeded³⁹²;
- OnRVFPictureNeeded³⁹³;
- OnControlAction³⁷³ (ControlAction=rvcaAfterRVFLoad);
- OnStyleTemplatesChange⁴¹⁰.

See also methods:

- InsertRVFFromStream³¹⁶;
- LoadRVFFromStream³²⁹;
- LoadRVF³²⁸;
- Format²⁸⁸;
- FormatTail²⁸⁸.

See also:

- TRichView component editor²¹⁸;
- Saving and loading¹²²;
- RVF overview¹²⁴;
- Paragraphs⁹⁵.

6.1.1.3.15 TRichView.AssignSoftPageBreaks

Displays soft page breaks (i.e. page breaks that were made automatically)

```
procedure AssignSoftPageBreaks(RVPrint759: TComponent);
```

(introduced in version 1.7)

Page breaks become visible after calling RichView.AssignSoftPageBreaks(RVPrint). **RVPrint** must be formatted (i.e. FormatPages⁷⁷⁴ must be called before this method). Page breaks are visible only if *rvShowPageBreaks* is included in RichView.Options²⁴⁰.

In editor⁴⁶¹, soft page breaks will be automatically cleared after any editing operation (any action invoking OnChange⁵⁷²). Page breaks will also be cleared when you call Clear²⁷⁹. In all other cases, you must call ClearSoftPageBreaks²⁷⁹ after each change in the document performed after calling AssignSoftPageBreaks.

Soft page breaks are drawn using RVStyle.SoftPageBreakColor⁶⁵¹ (or in OnDrawPageBreak⁶⁶⁸ event).

See also:

- ClearSoftPageBreaks²⁷⁹.

See also properties:

- PageBreaksBeforeItems²⁴⁴.

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⁽¹¹⁸⁾ of inserted controls⁽¹⁶⁸⁾. (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(348)(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(517)(' ', btn, rvvaBaseline);
```

4) Events are not saved in RVF, so reassign them on loading. Use OnControlAction⁽³⁷³⁾ 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 ⁽⁸⁵⁾;
- turns off displaying soft page breaks ⁽²⁷⁹⁾;
- clears DocProperties ⁽²³¹⁾;
- clears DocObjects ⁽²²⁹⁾;
- stops live spelling check ⁽²⁷⁹⁾;
- stops animations ⁽¹¹⁹⁾, if AnimationMode ⁽²²²⁾ = *rvaniOnFormat*.

This method does not change collections of text, paragraph and list styles. So, if your application adds styles while editing, call `DeleteUnusedStyles` ⁽²⁸⁶⁾ 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` ⁽³²⁸⁾, `LoadRVFFromStream` ⁽³²⁹⁾.

See also methods:

- `Format` ⁽²⁸⁸⁾.

See also method of TRichViewEdit:

- `Clear` ⁽⁵⁰¹⁾.

See also:

- `Methods for building documents` ⁽¹⁰²⁾.

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` ⁽²⁷⁹⁾ method calls this procedure automatically.

See also: `Live spelling check in RichView` ⁽¹³²⁾.

6.1.1.3.19 TRichView.ClearSoftPageBreaks

Turns off displaying soft page breaks

```
procedure ClearSoftPageBreaks;
```

See also:

- `AssignSoftPageBreaks` ⁽²⁷⁷⁾.

6.1.1.3.20 TRichView.ClientToDocument, DocumentToClient

The methods convert client coordinates to document coordinates and vice versa.

```
function ClientToDocument(const APoint: TRVCoordPoint998): TRVCoordPoint998;
(introduced in v1.9)
```

```
function DocumentToClient(const APoint: TRVCoordPoint998): TRVCoordPoint998;
(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, GetItemAt²⁹⁷ uses this coordinate system.

The methods take a cell rotation⁹⁰⁴ into account.

Note:

This method must be used instead of adding HScrollPos⁸¹⁶ to X and VScrollPos⁸¹⁹*VSmallStep²⁵⁷ to Y. The old method was used in some old example code, it is correct only if the document is top-aligned²⁵⁷.

See also:

Example in OnRVMouseDown³⁹⁴.

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: TRVStyleUnits1027);
```

(introduced in version 13)

The methods convert all lengths in the current document from Style²⁵³.Units⁶⁵⁵ to different units of measurement:

- pixels (1 pixel = 1/Style²⁵³.UnitsPixelsPerInch⁶⁵⁵ of an inch)
- twips,
- EMU
- NewUnits (pixels, twips, or EMU),

respectively.

The methods convert only properties of items¹⁵⁸, they do not convert properties of the Style²⁵³ itself.

To convert the document completely:

- call one of these methods (for all documents linked to the same TRVStyle⁶³⁰ component);
- call one of Style²⁵³.ConvertTo*⁶⁵⁷ method to convert properties of Style²⁵³ to the same units of measurement.

See also:

- Units of measurement in TRichView ⁽¹³⁹⁾

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.

See also:

- Working with selection ⁽¹⁰⁷⁾;
- Working with Clipboard ⁽¹⁰⁸⁾.

6.1.1.3.23 TRichView.CopyDef

Copies the selected fragment to the Clipboard in several formats according to Options ⁽²⁴⁰⁾.

function CopyDef: Boolean;

This is the main method for copying to the Clipboard. It is executed automatically when user presses **Ctrl + C** or **Ctrl + Insert**.

Calling CopyDef does the same thing as sending WM_COPY message to the control.

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

- if (*rvoAutoCopyRVF* in Options), it copies selection as RVF ⁽¹²⁴⁾.
- 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 ⁽¹³⁰⁾ text.
- if (*rvoAutoCopyRTF* in Options), it copies selection as RTF.

This method must be called only when the document is formatted.

Return value:

"Is the selection not empty?" and "Is at least one of *rvoAutoCopy**** options set?"

See also:

- Working with selection ⁽¹⁰⁷⁾;
- Working with Clipboard ⁽¹⁰⁸⁾.

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 ⁽¹⁰⁷⁾;
- Working with Clipboard ⁽¹⁰⁸⁾.

6.1.1.3.25 TRichView.CopyRTF

Copies the selected part of the document to the Clipboard as RTF (Rich Text Format).

procedure CopyRTF;

This procedure does nothing if there is nothing selected.

This procedure clears Clipboard before copying.

Settings for RTF saving can be changed in the TRichView component editor ⁽²¹⁸⁾.

Options for saving are in RTFOptions ⁽²⁴⁵⁾ property.

By default, items tags ⁽⁹¹⁾ are saved as hyperlinks targets. You can customize saving using OnWriteHyperlink ⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in OnWriteObjectProperties ⁽⁴¹³⁾ event.

This method must be called only when the document is formatted.

See also:

- SaveRTF ⁽³⁴³⁾, SaveRTFToStream ⁽³⁴⁴⁾ methods;
- RTFOptions ⁽²⁴⁵⁾ property;
- UseStyleTemplates ⁽²⁵⁶⁾ property;
- Working with selection ⁽¹⁰⁷⁾;
- Working with Clipboard ⁽¹⁰⁸⁾.

6.1.1.3.26 TRichView.CopyRVF

Copies the selection to the Clipboard as RVF (RichView Format)

procedure CopyRVF;

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

Settings for RVF saving can be changed in the TRichView component editor ⁽²¹⁸⁾.

This method must be called only when the document is formatted.

See also:

- Working with selection ⁽¹⁰⁷⁾;
- Working with Clipboard ⁽¹⁰⁸⁾;
- RVF ⁽¹²⁴⁾.

6.1.1.3.27 TRichView.CopyText -W

These methods copy the selected fragment to the Clipboard as text.

```
procedure CopyText;
```

```
procedure CopyTextA;
```

```
procedure CopyTextW;
```

These methods do nothing if there is nothing selected.

These methods clear the Clipboard before copying.

CopyTextA copies text in ANSI encoding. Text of Unicode styles is converted basing on TRVStyle.DefCodePage⁽⁶³⁵⁾.

CopyTextW and **CopyText** copy text in Unicode encoding.

Since modern versions of Windows converts Unicode and ANSI texts in the Clipboard to each other, using **CopyTextA** does not make sense (and it is not available in non-Windows FireMonkey version).

These methods must be called only when the document is formatted.

See also:

- Working with selection⁽¹⁰⁷⁾;
- Working with Clipboard⁽¹⁰⁸⁾;
- Unicode⁽¹³⁰⁾.

6.1.1.3.28 TRichView.Create

Usual component constructor. Creates a new TRichView instance.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate RichView at runtime. RichViews placed on forms at design time are created automatically. Specify the owner of the new RichView using the **AOwner** parameter.

Important note: initial values of several properties of components created in code may be different from initial values of components placed on form at design time. In the latter case, initial values of these properties can be overridden by the designtime component editor⁽²¹⁸⁾ (and overridden by default!)

6.1.1.3.29 TRichView.DeleteItems

Removes the specified number of items (**Count**) from the document, starting from the **FirstItemNo**-th item.

```
procedure DeleteItems(FirstItemNo, Count: Integer);
```

Items are indexed from 0 to ItemCount⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.DeleteItems.

Method type:  viewer-style.

Be careful: this method can produce invalid document⁽¹³⁸⁾.

Example 1: deleting all empty lines

```
procedure DeleteBlankLines(RVData: TCustomRVData(957));  
var i,r,c: Integer;
```

```

    table := TRVTableItemInfo846;
begin
  for i := RVData.ItemCount237 - 1 downto 0 do
    if RVData.IsParaStart318(i) and (RVData.GetItemStyle302(i) >= 0) and
      (RVData.GetItemTextR(i) = '') and (RVData.ItemCount237 > 1) then
      RVData.DeleteItems(i, 1)
    else if RVData.GetItem296(i) is TRVTableItemInfo846 then
      begin
        table := TRVTableItemInfo846(RVData.GetItem296(i));
        for r := 0 to table.RowCount864 - 1 do
          for c := 0 to table.ColCount858 - 1 do
            if table.Cells857[r, c] <> nil then
              DeleteBlankLines(table.Cells857[r, c].GetRVData907);
          end;
        end;
      end;
end;

```

DeleteBlankLines cannot be undone/redone. If called for editor, the editor's undo buffer becomes incorrect, so call ClearUndo⁵⁰¹.

Call:

```

DeleteBlankLines(RichViewEdit1.RVData247);
RichViewEdit1.ClearUndo501;
RichViewEdit1.Format288;

```

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;
end;

```

See also:

- DeleteSection²⁸⁵;
- DeleteParas²⁸⁵.

6.1.1.3.30 TRichView.DeleteMarkedStyles

Deletes unused text⁶⁵⁴, paragraph⁶⁴⁸ and list⁶⁴⁶ styles.

```

procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData969);

```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents¹²⁴ (but stored in file or in the Registry). In this case, several documents share the same collections of styles. Used styles are marked with MarkStylesInUse³³¹ method, see TRVDeleteUnusedStylesData⁹⁶⁹ 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⁽²⁸⁶⁾ instead.

See also methods:

- MarkStylesInUse⁽³³¹⁾;
- DeleteUnusedStyles⁽²⁸⁶⁾.

6.1.1.3.31 TRichView.DeleteParas

Deletes paragraph(s) containing items from **FirstItemNo** to **LastItemNo**, updates the document

```
procedure DeleteParas(FirstItemNo, LastItemNo: Integer);
```

(introduced in version 1.7)

This method is designed specially for chat/log windows. Together with FormatTail⁽²⁸⁸⁾, it allows updating chat windows faster, without complete reformatting.

Items are indexed from 0 to ItemCount⁽²³⁷⁾-1..

Method type: unique method. Unlike all other methods of TRichView, it quickly reformats TRichView, so you do not need to call Format⁽²⁸⁸⁾ 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⁽²⁸³⁾,
- DeleteSection⁽²⁸⁵⁾.

6.1.1.3.32 TRichView.DeleteSection

Removes the "section" named **CpName** from documents

```
procedure DeleteSection(CpName: String);
```

"Section" is a part of document. The first item of "section" is the item having *checkpoint*⁽⁸⁷⁾ 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⁽¹³⁸⁾.

See also:

- DeleteItems⁽²⁸³⁾;
- DeleteParas⁽²⁸⁵⁾.

6.1.1.3.33 TRichView.DeleteUnusedStyles

Deletes text and/or paragraph and/or list styles which do not exist in document

```
procedure DeleteUnusedStyles(  
    TextStyles, ParaStyles, ListStyles: Boolean);
```

(introduced in version 1.5, updated in version 1.6, 1.7)

This procedure deletes unused items from the collections of text (if **TextStyles** parameter is True), paragraph (if **ParaStyles** parameter is True), and list (if **ListStyles** parameter is True) styles (see TextStyles⁶⁵⁴, ParaStyles⁶⁴⁸, ListStyles⁶⁴⁶) of the linked²⁵³ 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 Standard⁶⁹⁵ property of styles)

This method does not require reformatting.

See also methods:

- MarkStylesInUse³³¹;
- DeleteMarkedStyles²⁸⁴.

6.1.1.3.34 TRichView.Deselect

Removes the document selection (does not change document, just nothing becomes selected for copying to the Clipboard)

```
procedure Deselect;
```

This method does not repaint document (call Invalidate)

This method must be called only when the document is formatted.

The method generates OnSelect⁴⁰⁸ event.

See also methods:

- SetSelectionBounds³⁶⁵;
- SelectAll³⁴⁸.

See also:

- Selecting in RichView document¹⁰⁷.

6.1.1.3.35 TRichView.Destroy

Destroys an instance of TRichView

```
destructor Destroy; override;
```

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

6.1.1.3.36 TRichView.FindCheckpointByName

Returns the first *checkpoint* having the specified **Name**, or **nil** if there is no such *checkpoint*

```
function FindCheckpointByName(  
    const Name: TRVUnicodeString(1032)): TCheckpointData(993);
```

See also methods:

- FindCheckpointByTag⁽²⁸⁷⁾;

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.1.3.37 TRichView.FindCheckpointByTag

Returns the first checkpoint having the specified tag, or nil if there is no such checkpoint

```
function FindCheckpointByTag(const Tag: TRVTag(1029)): TCheckpointData(993);
```

Example:

```
var  
    CheckpointData: TCheckpointData;  
    ...  
    CheckpointData := MyRichView.FindCheckpointByTag(  
        'Tag_String_Example');
```

See also methods:

- FindCheckpointByName⁽²⁸⁷⁾.

See also:

- "Tags"⁽⁹¹⁾;
- "Checkpoints"⁽⁸⁷⁾.

6.1.1.3.38 TRichView.FindControlItemNo

Returns an index of the RichView item which owns the control⁽¹⁶⁸⁾ **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⁽⁸⁵⁾.

6.1.1.3.39 TRichView.Format

Formats the document. Only formatted documents can be displayed or edited.

procedure `Format`;

If `rvoFormatInvalidate` in `Options`⁽²⁴⁰⁾ (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`⁽⁴⁶¹⁾ 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`⁽²⁸⁸⁾;
- `Reformat`⁽³³²⁾.

See also properties:

- `Options`⁽²⁴⁰⁾.

See also:

- Building document⁽¹⁰²⁾.

6.1.1.3.40 TRichView.FormatTail

Formats a part of document appended after the last call of `Format`⁽²⁸⁸⁾ 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`⁽²⁸⁸⁾ 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`⁽²⁸⁵⁾, it allows updating chat windows faster, without complete reformatting.

If `rvoFormatInvalidate` in `Options`⁽²⁴⁰⁾, this method also repaints RichView.

See also methods:

- `Format`⁽²⁸⁸⁾;
- `DeleteParas`⁽²⁸⁵⁾.

See also properties:

- `Options`⁽²⁴⁰⁾;
- `VScrollVisible`⁽⁸¹⁹⁾.

See also:

- Appending items⁽¹⁰²⁾.

6.1.1.3.41 TRichView.GetBreakInfo

Returns main properties of item of *break* type ⁽¹⁶⁷⁾

```
procedure GetBreakInfo(ItemNo: Integer; out AWidth: TRVStyleLength(1027);
  out AStyle: TRVBreakStyle(995); out AColor: TRVColor(996);
  out ATag: TRVTag(1029));
```

Input parameter:

ItemNo – index of the item. The item must be of *break* ⁽¹⁶⁷⁾ type (*rvsBreak* ⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError* ⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount* ⁽²³⁷⁾ -1, *GetItemStyle* ⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells ⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData* ⁽⁹⁰⁷⁾.*GetBreakInfo*.

Output parameters:

AWidth – line width (or rectangle height). This value is measured in *Style* ⁽²⁵³⁾.*Units* ⁽⁶⁵⁵⁾.

AStyle – visual style of this *break*, see *TRVBreakStyle* ⁽⁹⁹⁵⁾ for possible values.

AColor – line color. If it is equal to *rvclNone* ⁽¹⁰³⁸⁾, *Style* ⁽²⁵³⁾.*TextStyles* ⁽⁶⁵⁴⁾[0].*Color* ⁽⁷⁰¹⁾ is used.

ATag – *tag* of the item. Use *SetItemTag* ⁽³⁶⁰⁾ or *SetBreakInfo* ⁽³⁵¹⁾ to change *tag* of item.

Instead of this method, you can use *GetItemExtraIntPropertyEx* ⁽³⁰⁰⁾ and *GetItemTag* ⁽³⁰²⁾ methods.

Additional item properties are returned by the methods *GetItemExtraIntProperty* ⁽³⁰⁰⁾ and *GetItemExtraStrProperty* ⁽³⁰⁰⁾.

See also methods:

- *SetBreakInfo* ⁽³⁵¹⁾;
- *GetItemStyle* ⁽³⁰²⁾;
- *GetItemExtraIntProperty* ⁽³⁰⁰⁾;
- *GetItemExtraStrProperty* ⁽³⁰⁰⁾.

See also properties:

- *ItemCount* ⁽²³⁷⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾;
- Item types ⁽⁸⁵⁾;
- *Tags* ⁽⁹¹⁾.

6.1.1.3.42 TRichView.GetBulletInfo

Returns main properties of item of *bullet* type ⁽¹⁷⁰⁾

VCL and LCL:

```
procedure GetBulletInfo(ItemNo: Integer; out AName: TRVUnicodeString(1032);
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag(1029));
```

(changed in version 18)

FireMonkey:

```
procedure GetBulletInfo(ItemNo: Integer; out AName: TRVUnicodeString1032;
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag1029;
  out ImageWidth, ImageHeight: TRVStyleLength1027);
```

Input parameter:

ItemNo – index of the item. The item must be of *bullet*¹⁷⁰ or *hotspot*¹⁷² type (*rvsBullet*¹⁰⁵⁹ or *rvsHotspot*¹⁰⁵⁹), otherwise the method raises *ERichViewError*⁹⁵⁷ exception. Items are indexed from 0 to *ItemCount*²³⁷-1, *GetItemStyle*³⁰² returns type of item. Items of subdocuments (table cells⁸⁹⁵) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData*⁹⁰⁷.*GetBulletInfo*.

Output parameters:

AName – name of *bullet*¹⁷⁰. It can also be read using *GetItemText*³⁰³ method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using *GetItemExtraIntPropertyEx*³⁰⁰ method.

ATag – *tag* of the item. Use *SetItemTag*³⁶⁰ or *SetBulletInfo*³⁵² to change *tag* of item. This value can also be read using *GetItemTag*³⁰² 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*³⁰⁰ and *GetItemExtraStrProperty*³⁰⁰.

See also methods:

- *SetBulletInfo*³⁵²;
- *GetItemStyle*³⁰²;
- *GetItemExtraIntProperty*³⁰⁰;
- *GetItemExtraStrProperty*³⁰⁰.

See also properties:

- *ItemCount*²³⁷.

See also:

- Obtaining RichView items¹¹¹;
- Item types⁸⁵.
- *Tags*⁹¹.

6.1.1.3.43 TRichView.GetCheckpointByNo

Returns the **No**-th *checkpoint*.

```
function GetCheckpointByNo(No: Integer): TCheckpointData993;
```

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 cells⁸⁹⁵)

See also methods:

- GetCheckpointNo²⁹²;
- GetItemCheckpoint²⁹⁸.

See also:

- "Checkpoints"⁸⁷.

6.1.1.3.44 TRichView.GetCheckpointInfo

Returns information about the *checkpoint* **CheckpointData**

```
procedure GetCheckpointInfo(CheckpointData: TCheckpointData993;  
  out Tag: TRVTag1029; out Name: TRVUnicodeString1032;  
  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 OnCheckpointVisible³⁷² event for this *checkpoint*.

See also properties:

- CPEventKind²²⁷.

See also events:

- OnCheckpointVisible³⁷².

See also:

- "Checkpoints"⁸⁷;
- "Tags"⁹¹.

6.1.1.3.45 TRichView.GetCheckpointItemNo

Returns index of the RichView item which owns the checkpoint **CheckpointData**

```
function GetCheckpointItemNo(CheckpointData: TCheckpointData993): Integer;
```

Note: this method does not work with checkpoints in subdocuments (table cells⁸⁹⁵)

See also:

- Item types⁸⁵;

- "Checkpoints" ⁽⁸⁷⁾.

6.1.1.3.46 TRichView.GetCheckpointNo

Returns index of the *checkpoint* **CheckpointData**

```
function GetCheckpointNo(CheckpointData: TCheckpointData (993)): 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 ⁽⁸⁹⁵⁾)

See also methods:

- GetCheckpointByNo ⁽²⁹¹⁾;
- GetCheckpointItemNo ⁽²⁹¹⁾.

See also:

- "Checkpoints" ⁽⁸⁷⁾.

6.1.1.3.47 TRichView.GetCheckpointXY

Returns **X** and **Y** coordinates of the *checkpoint* (relative the top left corner of the document, i.e. top left corner of scrollable area)

```
procedure GetCheckpointXY(CheckpointData: TCheckpointData (993);  
  out X, Y: TRVCoord (998));
```

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 ⁽²³³⁾).

This method must be called only when the document is formatted.

See also methods:

- GetCheckpointY ⁽²⁹²⁾;
- GetCheckpointYEx ⁽²⁹³⁾;
- ScrollTo ⁽⁸²⁰⁾.

See also:

- "Checkpoints" ⁽⁸⁷⁾;
- Scrolling ⁽¹⁰⁵⁾.

6.1.1.3.48 TRichView.GetCheckpointY

Returns vertical coordinate of the **no**-th *checkpoint*

```
function GetCheckpointY(no: Integer): TRVCoord (998);
```

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⁽²⁹³⁾;
- GetCheckpointXY⁽²⁹²⁾;
- ScrollTo⁽⁸²⁰⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾;
- Vertical scrolling⁽¹⁰⁵⁾.

6.1.1.3.49 TRichView.GetCheckpointYEx

Returns vertical coordinate of the *checkpoint* **CheckpointData**

```
function GetCheckpointYEx(
    CheckpointData: TCheckpointData(993)): TRVCoord(998);
```

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⁽²⁹²⁾;
- GetCheckpointXY⁽²⁹²⁾;
- ScrollTo⁽⁸²⁰⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾;
- Vertical scrolling⁽¹⁰⁵⁾.

6.1.1.3.50 TRichView.GetControlInfo

Returns main properties of the item of "inserted control" type⁽¹⁶⁸⁾

```
procedure GetControlInfo(ItemNo: Integer;
    out AName: TRVUnicodeString(1032); out Actrl: TControl;
    out AAlign: TRVAlign(1033); out ATag: TRVTag(1029));
```

(changes in version 18)

Input parameter:

ItemNo – index of the item. The item must be of "inserted control" type⁽¹⁶⁸⁾ (*rvcComponent*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount*⁽²³⁷⁾-1, *GetItemStyle*⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData*⁽⁹⁰⁷⁾.*GetControlInfo*.

Output parameters:

AName – name of control item⁽¹⁶⁸⁾. Do not confuse with **Actrl.Name** property! This value can also be read using **GetItemText**⁽³⁰³⁾ 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**⁽³⁶⁰⁾ or **SetControllInfo**⁽³⁵⁴⁾ to change *tag* of item. This value can also be read using **GetItemTag**⁽³⁰²⁾ method.

Additional item properties are returned by the methods **GetItemExtraIntProperty**⁽³⁰⁰⁾ and **GetItemExtraStrProperty**⁽³⁰⁰⁾.

See also methods:

- **SetControllInfo**⁽³⁵⁴⁾;
- **GetItemStyle**⁽³⁰²⁾;
- **GetItemExtraIntProperty**⁽³⁰⁰⁾;
- **GetItemExtraStrProperty**⁽³⁰⁰⁾.

See also properties:

- **ItemCount**⁽²³⁷⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.1.3.51 TRichView.GetFirstCheckpoint

Returns value identifying the first *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetFirstCheckpoint: TCheckpointData(993);
```

GetFirstCheckpoint + **GetNextCheckpoint**⁽³⁰⁷⁾ allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells⁽⁸⁹⁵⁾) are not included in *checkpoints* list of the document.

See also methods:

- **GetNextCheckpoint**⁽³⁰⁷⁾, contains example;
- **GetLastCheckpoint**⁽³⁰⁸⁾;
- **GetPrevCheckpoint**⁽³¹¹⁾;
- **GetCheckpointInfo**⁽²⁹¹⁾;
- **GetCheckpointXY**⁽²⁹²⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.1.3.52 TRichView.GetFocusedItem

Returns the item having keyboard focus.

```
procedure GetFocusedItem(out ARVData: TCustomRVFormattedData957;
out AItemNo: Integer);
```

There are two types of items which can have keyboard focus:

- **hypertext**⁹² (text or pictures)
- **inserted controls**¹⁶⁸.

If hypertext item has focus, pressing **Enter** invokes **OnJump**³⁸² event for the focused link.

This method returns the focused item (the **AItemNo**-th item of **ARVData** object).

ARVData can be **TRichView.RVData**²⁴⁷ or table cell¹⁹².

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**⁴⁶¹.

See also:

- **TabNavigation**²⁵⁵ property.

6.1.1.3.53 TRichView.GetHotspotInfo

Returns main properties for item of *hotspot* type¹⁷²

VCL and LCL:

```
procedure GetHotspotInfo(ItemNo: Integer; out AName: TRVUnicodeString1032;
out AImageIndex, AHotImageIndex: Integer;
out AImageList: TCustomImageList;
out ATag: TRVTag1029);
```

FireMonkey:

```
procedure GetHotspotInfo(ItemNo: Integer; out AName: TRVUnicodeString1032;
out AImageIndex, AHotImageIndex: Integer;
out AImageList: TCustomImageList;
out ATag: TRVTag1029;
out ImageWidth, ImageHeight: TRVStyleLength1027);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of *hotspot*¹⁷² type (*rvsHotspot*¹⁰⁵⁹), otherwise the method raises **ERichViewError**⁹⁵⁷ exception. Items are indexed from 0 to **ItemCount**²³⁷-1, **GetItemStyle**³⁰² returns type of item. Items of subdocuments (table cells⁸⁹⁵) are not included in the items range of the main document; for items in cells, use **Cell.GetRVData**⁹⁰⁷.**GetHotspotInfo**.

Output parameters:

AName – name of *hotspot*¹⁷². It can also be read using **GetItemText**³⁰³ method.

AImageList – image list (a reference to image list, do not free it).

AlmageIndex – index in **AlmageList** for normal image. It can also be read using `GetItemExtraIntPropertyEx`⁽³⁰⁰⁾ method.

AHotImageIndex – index in **AlmageList** 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`⁽³⁰⁰⁾ method.

ATag – *tag* of the item. Use `SetItemTag`⁽³⁶⁰⁾ or `SetHotspotInfo`⁽³⁵⁷⁾ to change *tag* of item. This value can also be read using `GetItemTag`⁽³⁰²⁾ 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`⁽³⁰⁰⁾ and `GetItemExtraStrProperty`⁽³⁰⁰⁾.

See also methods:

- `SetHotspotInfo`⁽³⁵⁷⁾;
- `GetItemStyle`⁽³⁰²⁾;
- `GetItemExtraIntProperty`⁽³⁰⁰⁾;
- `GetItemExtraStrProperty`⁽³⁰⁰⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also:

- Obtaining `RichView` items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- *Tags*⁽⁹¹⁾.

6.1.1.3.54 TRichView.GetItem

Returns object representing one item in document by its index **ItemNo** (zero-based)

function `GetItem(ItemNo: Integer): TCustomRVItemInfo`⁽⁸⁴⁴⁾;

(introduced in version 1.4)

Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`GetItem`.

Do not free this object yourself.

Usually this method is used to get an item object for tables⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, references to notes⁽¹⁸⁴⁾.

This method is inverse to `GetItemNo`⁽³⁰¹⁾.

See also methods:

- `AddItem`⁽²⁶⁹⁾;

- `GetItemStyle` ⁽³⁰²⁾.

See also methods of TRichViewEdit:

- `GetCurrentItem` ⁽⁵⁰⁸⁾;
- `GetCurrentItemEx` ⁽⁵⁰⁹⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also events:

- `OnItemAction` ⁽³⁸⁰⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾.

6.1.1.3.55 TRichView.GetItemAt

Returns item ⁽¹⁵⁸⁾ at the position **X,Y**.

```
function GetItemAt(X, Y: TRVCoord (998) ;  
    out RVData: TCustomRVFormattedData (957) ;  
    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 ⁽²⁴⁷⁾ or table cell ⁽⁸⁹⁵⁾ 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` ⁽³¹⁴⁾.

See also events:

- OnRVMouseDown⁽³⁹⁴⁾ (contains example);
- OnRVMouseUp⁽³⁹⁶⁾.

Demo projects:

- Demos*\Assorted\PlanetQuiz\

6.1.1.3.56 TRichView.GetItemCheckpoint

Returns *checkpoint* associated with the **ItemNo**-th item, or **nil** if this item does not have associated *checkpoint*.

```
function GetItemCheckpoint(ItemNo: Integer): TCheckpointData(993);
```

Items are indexed from 0 to ItemCount⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.GetItemCheckpoint.

See also methods:

- GetCheckpointInfo⁽²⁹¹⁾;
- GetCheckpointXY⁽²⁹²⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.1.3.57 TRichView.GetItemCoords

The methods return coordinates of the item in RichView document.

```
function GetItemCoords(ItemNo: Integer;
```

```
  out Left,Top: TRVCoord(998)): Boolean;
```

```
function GetItemClientCoords(ItemNo: Integer;
```

```
  out Left,Top: TRVCoord(998)): Boolean;
```

(introduced in version 1.4)

Input parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽²⁹⁹⁾ does not have these limitations.

Return value:

True if coordinates were successfully retrieved (document must be formatted)

See also:

- ClientToDocument, DocumentToClient⁽²⁸⁰⁾.

See also methods of TCustomRVFormattedData

- GetOriginEx⁽⁹⁶⁰⁾.

6.1.1.3.58 TRichView.GetItemCoordsEx

Returns coordinates of the item in RichView document.

```
function GetItemCoordsEx(RVData: TCustomRVFormattedData(957);  
  ItemNo, Part: Integer; AllowFloating: Boolean;  
out R: TRVCoordRect(998)): Boolean;
```

(introduced in version 14)

Input parameters:

RVData, **ItemNo** define the position of the item. **RVData** – an object containing this item (RichView.RVData⁽²⁴⁷⁾ or table cell⁽⁸⁹⁵⁾ 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⁽¹⁰³³⁾ 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⁽²⁸⁰⁾.

The method must be called only when the document is formatted.

This method takes cell rotation⁽⁹⁰⁴⁾ into account.

Return value:

True if coordinates were successfully retrieved (document must be formatted; and **Part** must be valid).

See also:

- `GetItemCoords`⁽²⁹⁸⁾.

6.1.1.3.59 TRichView.GetItemExtraIntProperty -Ex

The methods return a value of the specified integer property for the **ItemNo**-th item.

```
function GetItemExtraIntProperty(ItemNo: Integer;
  Prop: TRVExtraItemProperty(1000); 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`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`GetItemExtraIntProperty[Ex]`.

Value receives value of the property identified by **Prop**.

In `GetItemExtraIntProperty`, **Prop**'s type is `TRVExtraItemProperty`⁽¹⁰⁰⁰⁾. 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⁽¹⁰⁵¹⁾

Return value:

True, if this item has this property. *False*, if not.

See also:

- `SetItemExtraIntProperty[Ex]`⁽³⁵⁸⁾;
- `GetItemExtraStrProperty`⁽³⁰⁰⁾.

See also methods of TRichViewEdit:

- `GetCurrentItemExtraIntProperty[Ex]`⁽⁵¹⁰⁾.

6.1.1.3.60 TRichView.GetItemExtraStrProperty -Ex

The methods return a value of the specified string property for the **ItemNo**-th item

```
function GetItemExtraStrProperty(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty(1005);
  out Value: TRVUnicodeString(1032)): Boolean;
function GetItemExtraStrPropertyEx(ItemNo, Prop: Integer;
  out Value: TRVUnicodeString(1032)): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.GetItemExtraStrProperty[Ex].

Value receives value of the property identified by **Prop**.

In **GetItemExtraStrProperty**, **Prop**'s type is TRVExtraltemStrProperty⁽¹⁰⁰⁵⁾. See information about this type for the list of properties.

In **GetItemExtraStrPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraltemStrProperty, **GetItemExtraStrPropertyEx** works like **GetItemExtraStrProperty**. In addition, it supports properties identified by rvespc*** constants⁽¹⁰⁵⁶⁾

Return value:

True, if this item has this property. *False*, if not.

See also:

- SetItemExtraStrProperty[Ex]⁽³⁵⁹⁾;
- GetItemExtraIntProperty[Ex]⁽³⁰⁰⁾.

See also methods of TRichViewEdit:

- GetCurrentItemExtraStrProperty[Ex]⁽⁵¹⁰⁾.

6.1.1.3.61 TRichView.GetItemNo

Returns index of item (zero-based).

```
function GetItemNo(Item: TCustomRVItemInfo(844)): Integer;
```

(introduced in version 1.4)

Item is an object representing the item. Usually this method if used to get index if table item⁽⁸⁴⁶⁾.

This method is inverse to GetItem⁽²⁹⁶⁾.

Returned value can be used, for example, in methods:

```
TRichViewEdit.BeginItemModify(495)/EndItemModify(503)
```

See also methods:

AddItem⁽²⁶⁹⁾.

6.1.1.3.62 TRichView.GetItemPara

Returns the index of paragraph style for paragraph containing the **ItemNo**-th item.

```
function GetItemPara(ItemNo: Integer): Integer;
```

(Introduced in version 1.3)

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.GetItemPara.

Return value:

The returned value is an index in the collection of `Style(253).ParaStyles(648)`. 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(318)`;
- `IsFromNewLine(317)`.

See also:

- `Paragraphs(95)`.

6.1.1.3.63 TRichView.GetItemStyle

Returns type (style) of the **ItemNo**-th item.

```
function GetItemStyle(ItemNo: Integer): Integer;
```

Parameters:

ItemNo – index of item (from 0 to `ItemCount(237)-1`). Items are indexed from 0 to `ItemCount(237)-1`. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData(907).GetItemStyle`.

Return value:

If the item is a text item⁽¹⁶¹⁾, this function returns zero or positive value: index of text style of this item in the collection of styles (`Style(253).TextStyles(654)`).

If the item is not a text item, this function returns type of this item. It is a negative value, see `rvs****` constants⁽¹⁰⁵⁹⁾.

See also properties of TRichViewEdit:

- `CurlItemStyle(473)`.

See also:

- `Item types(85)`.

6.1.1.3.64 TRichView.GetItemTag

Returns *tag* of the **ItemNo**-th item

```
function GetItemTag(ItemNo: Integer): TRVTag(1029);
```

Parameter:

ItemNo – index of the item. Items are indexed from 0 to `ItemCount(237)-1`. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData(907).GetItemTag`.

Use `SetItemTag(360)` to change tag of item.

See also methods:

- `SetItemTag(360)`.

See also method of TRichViewEdit:

- `GetCurrentTag`⁽⁵¹⁴⁾.

See also:

- RichView items' "tags"⁽⁹¹⁾.

6.1.1.3.65 TRichView.GetItemText -A -W

Returns a text of text item, or a name of non-text item.

```
function GetItemText(ItemNo: Integer): String;
```

```
function GetItemTextA(ItemNo: Integer): TRVAnsiString(993);
```

```
function GetItemTextW(ItemNo: Integer): TRVUnicodeString(1032);
```

(Introduced in version 1.3, 1.7)

ItemNo – index of the item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`.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⁽⁷⁰⁰⁾. If it is equal to `DEFAULT_CHARSET`, and for non-text items, `Style`⁽²⁵³⁾.`DefCodePage`⁽⁶³⁵⁾ 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⁽¹⁵⁴⁾

See also:

- `SetItemText -A -W`⁽³⁶⁰⁾.

See also method of TRichViewEdit:

- `GetCurrentItemText -A -W`⁽⁵¹¹⁾.

See also:

- Obtaining Items of RichView⁽¹¹¹⁾;
- Unicode in RichView⁽¹³⁰⁾.

6.1.1.3.66 TRichView.GetItemVAlign

Returns vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
function GetItemVAlign(ItemNo: Integer): TRVVAlign(1033);
```

(introduced in version 12)

Parameter:

ItemNo – index of the item. Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`.GetItemVAlign`.

This method must not be called for text items⁽¹⁶¹⁾, tables⁽¹⁷³⁾, *breaks*⁽¹⁶⁷⁾, tabs⁽¹⁶²⁾, list markers⁽¹⁷⁴⁾, footnotes⁽¹⁸⁰⁾ and endnotes⁽¹⁷⁸⁾.

Use `SetItemVAlign`⁽³⁶¹⁾ to change vertical alignment.

See also methods:

- SetItemVAlign⁽³⁶¹⁾.

See also method of TRichViewEdit:

- GetCurrentItemVAlign⁽⁵¹²⁾

6.1.1.3.67 TRichView.GetJumpPointItemNo

Returns index of hypertext item (*hotspot*⁽¹⁷²⁾, *hot-picture*⁽¹⁶⁶⁾ or text hypertext) by its hypertext **id**.

```
function GetJumpPointItemNo(id: Integer): Integer;
```

Warning: do not use this function for document containing tables⁽¹⁹⁰⁾. 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⁽³⁰⁴⁾ instead.

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo⁽²³⁶⁾.

Return value

Index of the hypertext item (in range 0..ItemCount⁽²³⁷⁾-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⁽⁴⁶¹⁾, if ReadOnly⁽⁴⁸⁰⁾=False, this function always returns -1 (except for calling it from inside OnWriteHyperlink⁽⁴¹¹⁾, OnJump⁽³⁸²⁾ or OnRVMouseMove⁽³⁹⁵⁾).

See also:

- RichView hypertext⁽⁹²⁾.

6.1.1.3.68 TRichView.GetJumpPointLocation

Returns information required to access item by its hypertext **id**.

```
procedure GetJumpPointLocation(id: Integer;  
  out RVData: TCustomRVFormattedData(957); out ItemNo: Integer);
```

(introduced in version 1.4)

Input parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo⁽²³⁶⁾.

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⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, 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 TCustomRichViewEdit⁴⁶¹, if ReadOnly⁴⁸⁰=False, this function always returns -1 (except for calling it from inside OnWriteHyperlink⁴¹¹, OnJump³⁸² or OnRVMouseMove³⁹⁵).

See also:

- RichView hypertext⁹².

6.1.1.3.69 TRichView.GetJumpPointY

Returns vertical coordinate of the hypertext link (*hotspot*¹⁷², *hot-picture*¹⁶⁶ or text hypertext), identified by **id**

```
function GetJumpPointY(id: Integer): TRVCoord998;
```

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo²³⁶.

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 TCustomRichViewEdit⁴⁶¹, if ReadOnly⁴⁸⁰=False, this function always returns 0 (except for calling it from inside OnJump³⁸² or OnRVMouseMove³⁹⁵).

See also methods:

- ScrollTo⁸²⁰

See also:

- RichView Hypertext⁹²

6.1.1.3.70 TRichView.GetLastCheckpoint

Returns value identifying the last *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetLastCheckpoint: TCheckpointData993;
```

(introduced in version 1.5)

GetLastCheckpoint + **GetPrevCheckpoint**⁽³¹¹⁾ 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**⁽³⁰⁷⁾, contains example;
- **GetFirstCheckpoint**⁽²⁹⁴⁾;
- **GetPrevCheckpoint**⁽³¹¹⁾;

- **GetCheckpointInfo**⁽²⁹¹⁾;
- **GetCheckpointXY**⁽²⁹²⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.1.3.71 TRichView.GetLineNo

Returns line of the position specified by **ItemNo** and **ItemOffs**.

```
function GetLineNo(ItemNo, ItemOffs: Integer): Integer;
```

(introduced in version 1.7)

Parameters

ItemNo – index of the item, in range from 0 to **ItemCount**⁽²³⁷⁾-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**⁽³⁰⁸⁾(**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(237)-1, GetOffsAfterItem(308)(ItemCount(237)-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`⁽³⁰²⁾.

See also methods of TCustomRichViewEdit:

- `GetCurrentLineCol`⁽⁵¹²⁾.

6.1.1.3.72 TRichView.GetListMarkerInfo

Returns main properties of list marker⁽¹⁷⁴⁾ (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`⁽²³⁷⁾-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⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`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`⁽²⁵³⁾.`ListStyles`⁽⁶⁴⁶⁾).

AListLevel – list level. This is an index in the collection of list levels (`Style.ListStyles[AListNo].Levels`⁽⁷³²⁾).

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`⁽³⁶²⁾.

6.1.1.3.73 TRichView.GetNextCheckpoint

Returns value identifying the next *checkpoint* after the given **CheckpointData**, or *nil* if **CheckpointData** is the last *checkpoint* in the document.

```
function GetNextCheckpoint (
  CheckpointData: TCheckpointData(993)): TCheckpointData(993);
```

`GetFirstCheckpoint`⁽²⁹⁴⁾ + **GetNextCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells⁸⁹⁵) are not included in *checkpoints* list of the document.

Example:

```
var Tag: TRVTag1029;
    Name: String;
    CheckpointData: TCheckpointData;
    RaiseEvent: Boolean;
    s: TRVUnicodeString1032;
begin
    MyListBox.Items.Clear;
    CheckpointData := MyRichView.GetFirstCheckpoint294;
    while CheckpointData<>nil do
    begin
        MyRichView.GetCheckpointInfo291(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:

- GetFirstCheckpoint²⁹⁴;
- GetLastCheckpoint³⁰⁵;
- GetPrevCheckpoint³¹¹;

- GetCheckpointInfo²⁹¹;
- GetCheckpointXY²⁹².

See also:

- "Checkpoints"⁸⁷.

6.1.1.3.74 TRichView.GetOffsAfterItem

Returns offset after the **ItemNo**-th item in the document

```
function GetOffsAfterItem(ItemNo: Integer): Integer;
```

This method is useful for selecting part of document for copying in the Clipboard, or for moving caret to the specified position in editor.

ItemNo – index of the item. Items are indexed from 0 to ItemCount²³⁷-1, GetItemStyle³⁰² returns type of item. Items of subdocuments (table cells⁸⁹⁵) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁹⁰⁷.GetOffsAfterItem.

For text items, the method returns length of text + 1. The exception is empty text items formatted with style having EmptyWidth⁷⁰¹>0; for them, the position after the item is 2.

For non-text items, it returns 1.

Example for TRichViewEdit⁽⁴⁶¹⁾: moving caret to the end of document

```
var ItemNo, Offs: Integer;  
  
ItemNo := MyRichViewEdit.ItemCount-1;  
Offs := MyRichViewEdit.GetOffsAfterItem(ItemNo);  
MyRichViewEdit.SetSelectionBounds(365)(ItemNo, Offs, ItemNo, Offs);  
MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds⁽³⁶⁵⁾;
- GetOffsBeforeItem⁽³⁰⁹⁾.

See also properties of TRichView:

- ItemCount⁽²³⁷⁾.

See also properties of TRichViewEdit:

- CurlItemNo⁽⁴⁷²⁾;
- OffsetInCurlItem⁽⁴⁷⁹⁾.

See also:

- Working with selection⁽¹⁰⁷⁾;
- How to: move a caret to the beginning or to the end of document in TRichViewEdit⁽¹⁰⁶⁵⁾.

6.1.1.3.75 TRichView.GetOffsBeforeItem

Returns offset before the **ItemNo**-th item in the document

```
function GetOffsBeforeItem(ItemNo: Integer): Integer;
```

This method is useful for selecting a part of document for copying in the Clipboard, or for moving caret to the specified position in editor.

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVDData⁽⁹⁰⁷⁾.GetOffsBeforeItem.

For text items, the method returns 1 For non-text items, it returns 0.

Example for TRichViewEdit⁽⁴⁶¹⁾: moving caret to the beginning of document

```
var ItemNo, Offs: Integer;  
  
ItemNo := 0;  
Offs := MyRichViewEdit.GetOffsBeforeItem(ItemNo);  
MyRichViewEdit.SetSelectionBounds(365)(ItemNo, Offs, ItemNo, Offs);  
MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds⁽³⁶⁵⁾;
- GetOffsAfterItem⁽³⁰⁸⁾.

See also properties of TRichView:

- `ItemCount` ⁽²³⁷⁾.

See also properties of TRichViewEdit:

- `CurlItemNo` ⁽⁴⁷²⁾;
- `OffsetInCurlItem` ⁽⁴⁷⁹⁾.

See also:

- Working with selection ⁽¹⁰⁷⁾;
- How to: move a caret to the beginning or to the end of document in TRichViewEdit ⁽¹⁰⁶⁵⁾;
- Example how to move caret to the beginning of paragraph ⁽⁵⁹²⁾.

6.1.1.3.76 TRichView.GetPictureInfo

Returns main properties for item of picture ⁽¹⁶³⁾ or *hot-picture* ⁽¹⁶⁶⁾ type

```
procedure GetPictureInfo(ItemNo: Integer; out AName: TRVUnicodeString (1032);
  out Agr: TRVGraphic (970); out AVAlign: TRVAlign (1033);
  out ATag: TRVTag (1029));
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of picture ⁽¹⁶³⁾ or *hot-picture* ⁽¹⁶⁶⁾ type (*rvsPicture* ⁽¹⁰⁵⁹⁾), otherwise the method raises `ERichViewError` ⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to `ItemCount` ⁽²³⁷⁾-1, `GetItemStyle` ⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells ⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData` ⁽⁹⁰⁷⁾.
.GetPictureInfo.

Output parameters:

AName – name of the item. This value can also be read using `GetItemText` ⁽³⁰³⁾ 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` ⁽³⁶⁰⁾ or `SetPictureInfo` ⁽³⁶³⁾ to change *tag* of item. This value can also be read using `GetItemTag` ⁽³⁰²⁾ 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` ⁽³⁰⁰⁾ and `GetItemExtraStrProperty` ⁽³⁰⁰⁾.

See also methods:

- `SetPictureInfo` ⁽³⁶³⁾;
- `GetItemStyle` ⁽³⁰²⁾;
- `GetItemExtraIntProperty` ⁽³⁰⁰⁾;
- `GetItemExtraStrProperty` ⁽³⁰⁰⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾;
- Item types ⁽⁸⁵⁾;
- "Tags" ⁽⁹¹⁾.

6.1.1.3.77 TRichView.GetPrevCheckpoint

Returns value identifying the previous *checkpoint* before the given **CheckpointData**, or **nil** if **CheckpointData** is the first *checkpoint* in the document.

```
function GetPrevCheckpoint(  
    CheckpointData: TCheckpointData (993)): TCheckpointData (993);
```

(introduced in version 1.5)

GetLastCheckpoint ⁽³⁰⁵⁾ + **GetPrevCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells ⁽⁸⁹⁵⁾) are not included in *checkpoints* list of document.

See also methods:

- GetFirstCheckpoint ⁽²⁹⁴⁾;
- GetLastCheckpoint ⁽³⁰⁵⁾;
- GetNextCheckpoint ⁽³⁰⁷⁾;

- GetCheckpointInfo ⁽²⁹¹⁾;
- GetCheckpointXY ⁽²⁹²⁾.

See also:

- "Checkpoints" ⁽⁸⁷⁾.

6.1.1.3.78 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 ⁽²³³⁾ contains a positive value, this function returns it.

Otherwise, if RichViewPixelsPerInch ⁽¹⁰⁴⁷⁾ (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 $96 * \text{ZoomPercent} ⁽²⁵⁸⁾ / 100$.

6.1.1.3.79 TRichView.GetSelectedImage

Returns the selected image

```
function GetSelectedImage: TRVGraphic(970);
```

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⁽¹⁰⁷⁾.

6.1.1.3.80 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⁽¹⁶¹⁾, 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⁽⁷⁰¹⁾>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⁽⁷⁰¹⁾>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⁽⁹⁸⁴⁾ unit instead.

See also methods:

- SetSelectionBounds⁽³⁶⁵⁾.

See also:

- Working with selection.⁽¹⁰⁷⁾

6.1.1.3.81 TRichView.GetSelText -A -W

These methods return the selected part of the document as a text.

```
function GetSelText: String;  
function GetSelTextA: TRVAnsiString(993);  
function GetSelTextW: TRVUnicodeString(1032);
```

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⁽²⁵³⁾.DefCodePage⁽⁶³⁵⁾ 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⁽¹⁵⁴⁾

These methods must be called only when the document is formatted.

See also:

- Working with selection⁽¹⁰⁷⁾.

6.1.1.3.82 TRichView.GetTextInfo

Returns text and *tag* for the **ItemNo**-th item

```
procedure GetTextInfo(ItemNo: Integer; out AText: TRVUnicodeString(1032);  
    out ATag: TRVTag(1029));
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of text type (`GetItemStyle`³⁰² must return zero or positive value for this item), otherwise the method raises `ERichViewError`⁹⁵⁷ exception. Items are indexed from 0 to `ItemCount`²³⁷-1. Items of subdocuments (table cells⁸⁹⁵) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁹⁰⁷.`GetTextInfo`.

Output parameters:

AText – item text (visible text). This is the same value as returned by `GetItemText`³⁰³.

ATag – tag of the item. Use `SetItemTag`³⁶⁰ to change tag of item. The tag can also be returned by `GetItemTag`³⁰².

See also methods:

- `GetItemStyle`³⁰²;
- `GetItemText`³⁰³, `SetItemText`³⁶⁰;
- `GetItemTag`³⁰², `SetItemTag`³⁶⁰.

See also properties:

- `ItemCount`²³⁷.

See also:

- Obtaining items¹¹¹;
- Item types⁸⁵;
- "Tags"⁹¹.

6.1.1.3.83 TRichView.GetWordAt -A -W

Returns word at the specified coordinates

```

procedure GetWordAt(X,Y: TRVCoord998);
  out RVData: TCustomRVFormattedData957; out ItemNo: Integer;
  out Word: TRVUnicodeString1032); overload;
function GetWordAtA(X,Y: TRVCoord998): TRVAnsiString993;
function GetWordAtW(X,Y: TRVCoord998): TRVUnicodeString1032;
function GetWordAt(X,Y: TRVCoord998): 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 item¹⁶¹), 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`²²⁸ 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 `Style253.DefCodePage635`.

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 Lazarus¹⁵⁴

For example, you can use this method inside `OnRVMouseDown394` and `OnRVMouseUp396`.

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: TRVCoord998);
var
  LItemNo: Integer;
  LRVData: TCustomRVFormattedData;
  LWord: TRVUnicodeString1032;
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.GetItemStyle302(LItemNo) >= 0 then // text item?
    Label1.Caption := LWord;
end;
```

See also events:

- `OnRVMouseDown394`;
- `OnRVMouseUp396`;
- `OnRVRightClick397`;
- `OnRVDbClick391`.

See also properties:

- `Delimiters228`.

See also methods:

- `SelectWordAt349`

6.1.1.3.84 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⁽²³⁷⁾. Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽²¹⁸⁾.

If style templates are used⁽²⁵⁶⁾, and RVFTextStylesReadMode⁽²⁵¹⁾=RVFParaStylesReadMode⁽²⁵¹⁾=rvf_sInsertMerge, and StyleTemplateInsertMode⁽²⁵⁴⁾<>rvstimIgnoreSourceStyleTemplates, the method merges style templates from RVF into Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾, and reads text and paragraph styles according to StyleTemplateInsertMode⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange⁽⁴¹⁰⁾ 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(237));
  Stream.Free;
end;

...
RichView1.Clear(279);
RichView1.DeleteUnusedStyles(286)(True, True, True);
AddDoc('c:\file1.rvf', RichView1);
AddDoc('c:\file2.rvf', RichView1);
AddDoc('c:\file3.rvf', RichView1);
RichView1.Format(288);

```

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(237);
  Stream := TFileStream.Create(FileName, fmOpenRead);
  rv.InsertRVFFromStream(Stream, rv.ItemCount(237));
  Stream.Free;
  if (ItemCount>0) and (ItemCount<rv.ItemCount(237)) then
    rv.PageBreaksBeforeItems(244)[ItemCount] := True;
end;

```

See also properties:

- RVFOptions⁽²⁴⁷⁾;
- RVFWarnings⁽²⁵¹⁾;
- RVFTextStylesReadMode⁽²⁵¹⁾ (this method type: RVF insertion);
- RVFParaStylesReadMode⁽²⁵¹⁾ (this method type: RVF insertion).

See also events:

- OnRVFImageListNeeded⁽³⁹³⁾;
- OnRVFControlNeeded⁽³⁹²⁾;
- OnRVFPictureNeeded⁽³⁹³⁾;
- OnControlAction⁽³⁷³⁾ (ControlAction=rvcaAfterRVFLoad);
- OnStyleTemplatesChange⁽⁴¹⁰⁾.

See also methods:

- AppendRVFFromStream⁽²⁷⁶⁾;
- LoadRVFFromStream⁽³²⁹⁾;
- LoadRVF⁽³²⁸⁾;
- Format⁽²⁸⁸⁾;
- FormatTail⁽²⁸⁸⁾.

See also methods of TRichViewEdit:

- InsertRVFFromStreamEd⁽⁵²⁹⁾.
- **See also:**
- TRichView component editor⁽²¹⁸⁾;
- Saving and loading⁽¹²²⁾;
- RVF overview⁽¹²⁴⁾.

6.1.1.3.85 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⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.IsFromNewLine.

This method returns *True* both in cases when this item starts a new paragraph (see IsParaStart⁽³¹⁸⁾) 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⁽³⁵⁰⁾ (*False*)).

See also methods:

- IsParaStart⁽³¹⁸⁾,
- GetItemPara⁽³⁰¹⁾.

See also:

- Paragraphs⁽⁹⁵⁾.

6.1.1.3.86 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⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.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⁽³⁵⁰⁾ (*False*)).

See also methods:

- IsFromNewLine⁽³¹⁷⁾;
- GetItemPara⁽³⁰¹⁾.

See also:

- Paragraphs⁽⁹⁵⁾;
- Example how to move caret to the beginning of paragraph⁽⁵⁹²⁾.

6.1.1.3.87 TRichView.LiveSpellingValidateWord

Removes live spelling underlines from all occurrences of **AWord**.

procedure LiveSpellingValidateWord(const AWord: TRVUnicodeString⁽¹⁰³²⁾);

(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⁽⁷⁹²⁾ in TRVSpellChecker⁽⁷⁸⁴⁾ component, the spell-checker calls this method automatically.

See also:

- Live spelling check in RichView⁽¹³²⁾.

6.1.1.3.88 TRichView.LoadDocX

Appends the content of DocX (Microsoft Word Document) file **FileName** to the document.

function LoadDocX(const FileName: TRVUnicodeString⁽¹⁰³²⁾): 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⁽²⁷⁹⁾ method before loading (and maybe DeleteUnusedStyles⁽²⁸⁶⁾ after Clear)

Method type:  viewer-style.

Parameters for loading are in RTFReadProperties⁽²⁴⁶⁾. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode⁽⁴⁵⁸⁾ and ParaStyleMode⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

See important note on loading external images from DocX⁽¹⁰⁶⁹⁾.

By default, hyperlink targets are loaded in items tags⁽⁹¹⁾. You can customize loading using OnReadHyperlink⁽³⁸⁸⁾ event.

If style templates are used⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=rvsAddIfNeeded, and StyleTemplateInsertMode⁽²⁵⁴⁾ <rvstimIgnoreSourceStyleTemplates, the method loads a DocX style sheet into Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾. 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⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange⁽⁴¹⁰⁾ event.

Return value: "Was reading successful?" (extended information is in RTFReadProperties.DocXErrorCode⁽⁴⁵⁴⁾)

See also methods of TRichView:

- LoadDocXFromStream⁽³¹⁹⁾;
- SaveDocX⁽³³³⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾;
- OnAssignImageFileName⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- InsertDocXFromFileEd⁽⁵¹⁸⁾.

See also:

- Saving and loading in RichView⁽¹²²⁾.

6.1.1.3.89 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⁽²⁷⁹⁾ method before loading (and maybe DeleteUnusedStyles⁽²⁸⁶⁾ after Clear)

Method type:  viewer-style.

Parameters for loading are in TRichView.RTFReadProperties⁽²⁴⁶⁾. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode⁽⁴⁵⁸⁾ and ParaStyleMode⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

See important note on loading external images from RTF ⁽¹⁰⁶⁹⁾.

By default, hyperlink targets are loaded in items tags ⁽⁹¹⁾. You can customize loading using OnReadHyperlink ⁽³⁸⁸⁾ event.

If style templates are used ⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode ⁽²⁵⁴⁾ <>*rvstimIgnoreSourceStyleTemplates*, the method loads a DocX style sheet into Style ⁽²⁵³⁾.StyleTemplates ⁽⁶⁵²⁾. 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 ⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange ⁽⁴¹⁰⁾ event.

Return value: "Was reading successful?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode ⁽⁴⁵⁴⁾)

See also methods of TRichView:

- LoadDocX ⁽³¹⁸⁾;
- SaveDocXToStream ⁽³³⁴⁾.

See also events of TRichView:

- OnImportPicture ⁽³⁷⁸⁾;
- OnReadHyperlink ⁽³⁸⁸⁾;
- OnStyleTemplatesChange ⁽⁴¹⁰⁾;
- OnProgress ⁽³⁸⁵⁾.

See also methods of TRichViewEdit:

- InsertDocXFromStreamEd ⁽⁵¹⁸⁾.

See also:

- Saving and loading in RichView ⁽¹²²⁾.

6.1.1.3.90 TRichView.LoadFromFile -Ex

The methods load a document from file **FileName**, autodetecting its format (RVF ⁽¹²⁴⁾ /RTF/HTML/DocX/Markdown/text)

```
function LoadFromFile(const FileName: TRVUnicodeString1032;
  IsTextUnicode: TRVYesNoAuto1034;
  AllowMarkdown: Boolean = False;
  CodePage: TRVCodePage996 = RV_CP_DEFAULT1050): Boolean; Boolean;
function LoadFromFileEx(const FileName: TRVUnicodeString1032;
  IsTextUnicode: TRVYesNoAuto1034; AllowedFormats: TRVLoadFormats;
  out Unicode: Boolean;
  out LoadedFormat: TRVLoadFormat;
  CodePage: TRVCodePage996 = RV_CP_DEFAULT1050): Boolean;): Boolean;
```

(introduced in version 21, modified in version 23)

These methods are analogs of LoadFromStream and LoadFromStreamEx ⁽³²¹⁾.

Methods type:  viewer-style.

Return value:

"Was reading successful?"

See also methods of TRichView:

- LoadRVF⁽³²⁸⁾;
- LoadRTF⁽³²⁶⁾;
- LoadDocX⁽³¹⁸⁾;
- LoadMarkdown⁽³²⁵⁾;
- LoadHTML⁽³²²⁾;
- LoadText, LoadTextW⁽³³⁰⁾.

6.1.1.3.91 TRichView.LoadFromStream -Ex

The methods load a document from **Stream**, autodetecting its format (RVF⁽¹²⁴⁾ /RTF/HTML/DocX/Markdown/text)

```
function LoadFromStream(Stream: TStream; IsTextUnicode: TRVYesNoAuto(1034);
  AllowMarkdown: Boolean = False;
  const Path: TRVUnicodeString(1032) = '';
  CodePage: TRVCodePage(996) = RV_CP_DEFAULT(1050)): Boolean;
function LoadFromStreamEx(Stream: TStream;
  IsTextUnicode: TRVYesNoAuto(1034); AllowedFormats: TRVLoadFormats(1015);
  out Unicode: Boolean; out LoadedFormat: TRVLoadFormat(1015);
  const Path: TRVUnicodeString = '';
  CodePage: TRVCodePage(996) = RV_CP_DEFAULT(1050)): 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⁽¹³⁰⁾ (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⁽¹⁰⁵⁰⁾ (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⁽³²⁰⁾
- LoadRVFFromStream⁽³²⁹⁾;
- LoadRTFFromStream⁽³²⁷⁾;
- LoadDocXFromStream⁽³¹⁹⁾;
- LoadMarkdownFromStream⁽³²⁵⁾;
- LoadHTMLFromStream⁽³²⁴⁾;
- LoadTextFromStream, LoadTextFromStreamW⁽³³¹⁾.

6.1.1.3.92 TRichView.LoadHTML

Appends the content of HTML file **FileName** to the document.

```
function LoadHTML(const FileName: TRVUnicodeString(1032);  
CodePage: TRVCodePage(996) = 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⁽²⁷⁹⁾ method before loading (and maybe DeleteUnusedStyles⁽²⁸⁶⁾ 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⁽²³⁷⁾.

See important note on loading external images from HTML⁽¹⁰⁶⁹⁾.

By default, hyperlink targets are loaded in items tags⁽⁹¹⁾. You can customize loading using OnReadHyperlink⁽³⁸⁸⁾ 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⁽³²⁰⁾ method, it can detect and load UTF-16 HTML.

Style templates

If style templates are used⁽²⁵⁶⁾, and `StyleTemplateInsertMode`⁽²⁵⁴⁾ `<>rvstmlgnoreSourceStyleTemplates`, the method loads an HTML style sheet into `Style`⁽²⁵³⁾ `.StyleTemplates`⁽⁶⁵²⁾. 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`⁽²⁵⁴⁾. The method calls `OnStyleTemplatesChange`⁽⁴¹⁰⁾ 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
<i>p</i>	'Normal'	<i>rvstkPara</i>
<i>h1..h6</i>	'heading 1' .. 'heading 6'	<i>rvstkParaText</i>
<i>pre</i>	'HTML Preformatted'	
<i>address</i>	'HTML Address'	
<i>blockquote</i>	'Quote'	
<i>strong</i>	'Strong'	<i>rvstkText</i>
<i>em</i>	'Emphasis'	
<i>a</i>	'Hyperlink'	
<i>code</i>	'HTML Code'	
<i>acronym</i>	'HTML Acronym'	
<i>var</i>	'HTML Variable'	
<i>kbd</i>	'HTML Keyboard'	
<i>samp</i>	'HTML Sample'	
<i>tt</i>	'HTML Typewriter'	
<i>dfn</i>	'HTML Definition'	
<i>cite</i>	'HTML Cite'	

Return value: "Was reading successful?"

See also methods of `TRichView`:

- `LoadHTMLFromStream`⁽³²⁴⁾;
- `SaveHTML`⁽³³⁵⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnImportFile⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾;
- OnAssignImageFileName⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- InsertHTMLFromFileEd⁽⁵²¹⁾.

See also:

- Saving and loading in TRichView⁽¹²²⁾.

6.1.1.3.93 TRichView.LoadHTMLFromStream

Appends the content of HTML from **Stream**.

```
function LoadHTMLFromStream(Stream: TStream;
  const Path: TRVUnicodeString(1032) = '';
  CodePage: TRVCodePage(996) = CP_ACP): Boolean;
```

(introduced in version 21)

The same as LoadHTML⁽³²²⁾, but loads from a stream instead of a file.

Parameters

Stream – stream containing HTML

Path defines a base path for images. If HTMLReadProperties⁽²³⁷⁾.BasePathLinks⁽⁴²⁵⁾ = *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⁽²³⁷⁾.

See details in the topic about LoadHTML⁽³²²⁾.

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadHTML⁽³²²⁾;
- SaveHTMLToStream⁽³³⁸⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnImportFile⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾;
- OnAssignImageFileName⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- InsertHTMLFromStreamEd⁽⁵²¹⁾.

See also:

- Saving and loading in TRichView⁽¹²²⁾.

6.1.1.3.94 TRichView.LoadMarkdown

Appends the content of **Markdown** file **FileName** to the document.

```
function LoadMarkdown(const FileName: TRVUnicodeString(1032);  
    CodePage: TRVCodePage(996) = 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⁽²⁷⁹⁾ method before loading (and maybe DeleteUnusedStyles⁽²⁸⁶⁾ 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⁽²³⁹⁾. See the topic about the class of this property⁽⁴³⁹⁾ 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⁽⁹¹⁾. You can customize loading using OnReadHyperlink⁽³⁸⁸⁾ event.

Return value: "Was reading successful?"

See also methods of TRichView:

- LoadMarkdownFromStream⁽³²⁵⁾;
- SaveMarkdown⁽³⁴⁰⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnAssignImageFileName⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- InsertMarkdownFromFileEd⁽⁵²³⁾.

See also:

- Saving and loading in TRichView⁽¹²²⁾.

6.1.1.3.95 TRichView.LoadMarkdownFromStream

Appends the content of **Markdown** stream **Stream** to the document.

```
function LoadMarkdownFromStream(Stream: TStream;  
    const Path: TRVUnicodeString(1032);
```

```
CodePage: TRVCodePage = CP_UTF8(996)): 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`⁽²⁷⁹⁾ method before loading (and maybe `DeleteUnusedStyles`⁽²⁸⁶⁾ 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 `rvmdIoBasePathLinks` is included `MarkdownProperties`⁽²³⁹⁾.`LoadOptions`⁽⁴⁴⁶⁾, this path is also used for hyperlinks.

Method type:  viewer-style.

Parameters for loading are in `MarkdownProperties`⁽²³⁹⁾. See the topic about the class of this property⁽⁴³⁹⁾ 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⁽⁹¹⁾. You can customize loading using `OnReadHyperlink`⁽³⁸⁸⁾ event.

Return value: "Was reading successful?"

See also methods of TRichView:

- `LoadMarkdown`⁽³²⁵⁾;
- `SaveMarkdown`⁽³⁴⁰⁾.

See also events of TRichView:

- `OnImportPicture`⁽³⁷⁸⁾;
- `OnReadHyperlink`⁽³⁸⁸⁾;
- `OnAssignImageFileName`⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- `InsertMarkdownFromStreamEd`⁽⁵²⁴⁾.

See also:

- Saving and loading in `TRichView`⁽¹²²⁾.

6.1.1.3.96 TRichView.LoadRTF

Appends the content of RTF (Rich Text Format) file **FileName** to the document.

```
function LoadRTF(const FileName: TRVUnicodeString(1032)): 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`⁽²⁷⁹⁾ method before loading (and maybe `DeleteUnusedStyles`⁽²⁸⁶⁾ after `Clear`).

Method type:  viewer-style.

Parameters for loading are in `RTFReadProperties`⁽²⁴⁶⁾. The most important properties affecting RTF loading are `RTFReadProperties.TextStyleMode`⁽⁴⁵⁸⁾ and `ParaStyleMode`⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

See important note on loading external images from RTF⁽¹⁰⁶⁹⁾.

By default, hyperlink targets are loaded in items tags⁽⁹¹⁾. You can customize loading using OnReadHyperlink⁽³⁸⁸⁾ event.

If style templates are used⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=rvsAddIfNeeded, and StyleTemplateInsertMode⁽²⁵⁴⁾ <>rvstimIgnoreSourceStyleTemplates, the method loads an RTF style sheet into Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾. 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⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange⁽⁴¹⁰⁾ event.

Return value: "Was reading successful?" (extended information is in RTFReadProperties.RTFErrorCode⁽⁴⁵⁷⁾)

See also methods of TRichView:

- LoadRTFFromStream⁽³²⁷⁾;
- SaveRTF⁽³⁴³⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾;
- OnAssignImageFileName⁽³⁷¹⁾.

See also methods of TRichViewEdit:

- InsertRTFFromFileEd⁽⁵²⁷⁾.

See also:

- Saving and loading in RichView⁽¹²²⁾.

6.1.1.3.97 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⁽²⁷⁹⁾ method before loading (and maybe DeleteUnusedStyles⁽²⁸⁶⁾ after Clear)

Method type:  viewer-style.

Parameters for loading are in TRichView.RTFReadProperties⁽²⁴⁶⁾. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode⁽⁴⁵⁸⁾ and ParaStyleMode⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

See important note on loading external images from RTF ⁽¹⁰⁶⁹⁾.

By default, hyperlink targets are loaded in items tags ⁽⁹¹⁾. You can customize loading using `OnReadHyperlink` ⁽³⁸⁸⁾ event.

If style templates are used ⁽²⁵⁶⁾, and `RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=rvsAddIfNeeded`, and `StyleTemplateInsertMode` ⁽²⁵⁴⁾ `<>rvstimIgnoreSourceStyleTemplates`, the method loads an RTF style sheet into `Style` ⁽²⁵³⁾.`StyleTemplates` ⁽⁶⁵²⁾. 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` ⁽²⁵⁴⁾. The method calls `OnStyleTemplatesChange` ⁽⁴¹⁰⁾ event.

Return value: "Was reading successful?" (extended information is in `TRichView.RTFReadProperties.RTFErrorCode` ⁽⁴⁵⁷⁾)

See also methods of TRichView:

- `LoadRTF` ⁽³²⁶⁾;
- `SaveRTFToStream` ⁽³⁴⁴⁾.

See also events of TRichView:

- `OnImportPicture` ⁽³⁷⁸⁾;
- `OnReadHyperlink` ⁽³⁸⁸⁾;
- `OnStyleTemplatesChange` ⁽⁴¹⁰⁾;
- `OnProgress` ⁽³⁸⁵⁾.

See also methods of TRichViewEdit:

- `InsertRTFFromStreamEd` ⁽⁵²⁸⁾.

See also:

- Saving and loading in `RichView` ⁽¹²²⁾.

6.1.1.3.98 TRichView.LoadRVF

Clears the document and loads a new document from the file **FileName** as RVF (RichView Format) ⁽¹²⁴⁾).

```
function LoadRVF(const FileName: TRVUnicodeString (1032)): Boolean;
```

(changed in version 18)

Method type:  viewer-style.

Setting for RVF loading can be changed in the `TRichView` component editor ⁽²¹⁸⁾.

Return value: "Was reading successful?"

See also properties:

- `RVFOptions` ⁽²⁴⁷⁾.
- `RVFWarnings` ⁽²⁵¹⁾;
- `RVFTextStylesReadMode` ⁽²⁵¹⁾ (this method type: RVF loading);
- `RVFParaStylesReadMode` ⁽²⁵¹⁾ (this method type: RVF loading);
- `UseStyleTemplates` ⁽²⁵⁶⁾.

See also events:

- OnRVFImageListNeeded⁽³⁹³⁾;
- OnRVFControlNeeded⁽³⁹²⁾;
- OnRVFPictureNeeded⁽³⁹³⁾;
- OnControlAction⁽³⁷³⁾ (ControlAction=rvcaAfterRVFLoad);
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾.

See also methods:

- LoadRVFFromStream⁽³²⁹⁾.
- AppendRVFFromStream⁽²⁷⁶⁾;
- InsertRVFFromStream⁽³¹⁶⁾;
- Format⁽²⁸⁸⁾;
- FormatTail⁽²⁸⁸⁾.

See also methods of TRichViewEdit:

- InsertRVFFromFileEd⁽⁵²⁸⁾.
- **See also:**
- TRichView component editor⁽²¹⁸⁾;
- Saving and loading⁽¹²²⁾;
- RVF overview⁽¹²⁴⁾.

6.1.1.3.99 TRichView.LoadRVFFromStream

Clears the document and loads a new document from the file stream **Stream** as RVF (RichView Format⁽¹²⁴⁾).

```
function LoadRVFFromStream(Stream: TStream): Boolean;
```

Method type:  viewer-style.

Setting for RVF loading can be changed in the TRichView component editor⁽²¹⁸⁾.

Return value: "Was reading successful?"

See also properties:

- RVFOptions⁽²⁴⁷⁾;
- RVFWarnings⁽²⁵¹⁾;
- RVFTextStylesReadMode⁽²⁵¹⁾ (this method type: RVF loading);
- RVFParaStylesReadMode⁽²⁵¹⁾ (this method type: RVF loading);
- UseStyleTemplates⁽²⁵⁶⁾.

See also events:

- OnRVFImageListNeeded⁽³⁹³⁾;
- OnRVFControlNeeded⁽³⁹²⁾;
- OnRVFPictureNeeded⁽³⁹³⁾;
- OnControlAction⁽³⁷³⁾ (ControlAction=rvcaAfterRVFLoad);
- OnStyleTemplatesChange⁽⁴¹⁰⁾;
- OnProgress⁽³⁸⁵⁾.

See also methods:

- LoadRVF⁽³²⁸⁾;

- AppendRVFFromStream⁽²⁷⁶⁾;
- InsertRVFFromStream⁽³¹⁶⁾;
- Format⁽²⁸⁸⁾;
- FormatTail⁽²⁸⁸⁾.

See also methods of TRichViewEdit:

- InsertRVFFromFileEd⁽⁵²⁸⁾.
- **See also:**
- TRichView component editor⁽²¹⁸⁾;
- Saving and loading⁽¹²²⁾;
- RVF overview⁽¹²⁴⁾.

6.1.1.3.100 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(1032);
  StyleNo, ParaNo: Integer; AsSingleParagraph: Boolean;
  CodePage: Cardinal=CP_ACP):Boolean;
```

```
function LoadTextW(const FileName: TRVUnicodeString(1032);
  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⁽²⁷⁹⁾ method before loading (and may be DeleteUnusedStyles⁽²⁸⁶⁾ after Clear).

Parameters

StyleNo – an index of text style⁽⁶⁵⁴⁾ which will be used for new text. It defines font attributes for the added text.

ParaNo – an index of paragraph style⁽⁶⁴⁸⁾ 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⁽³⁵⁰⁾);
- *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 ⁽²⁵⁶⁾ 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 ⁽³³¹⁾;
- AddTextNLA, AddTextNLW ⁽²⁷⁴⁾;
- Format ⁽²⁸⁸⁾;
- FormatTail ⁽²⁸⁸⁾;
- SetAddParagraphMode ⁽³⁵⁰⁾;
- SaveText, SaveTextW ⁽³⁴⁵⁾.

See also methods of TRichViewEdit:

- InsertTextFromFile, InsertTextFromFileW ⁽⁵³³⁾.

See also properties of TRVStyle:

- TextStyles ⁽⁶⁵⁴⁾.
- SpacesInTab ⁽⁶⁵²⁾.

See also:

- Other methods for appending items ⁽¹⁰²⁾;
- Saving and loading ⁽¹²²⁾;
- Unicode in RichView ⁽¹³⁰⁾;
- Example how to load UTF-8 files ⁽⁴⁶⁰⁾.

6.1.1.3.101 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 ⁽³³⁰⁾, but read text from streams instead of files.

6.1.1.3.102 TRichView.MarkStylesInUse

Stores lists of used text ⁽⁶⁵⁴⁾, paragraph ⁽⁶⁴⁸⁾ and list ⁽⁶⁴⁶⁾ styles.

```
procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData (969));
```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents ⁽¹²⁴⁾ (but stored in file or in the Registry). In this case, several documents share the same collections of styles. See TRVDeleteUnusedStylesData ⁽⁹⁶⁹⁾ 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⁽²⁸⁶⁾ instead.

See also methods:

- DeleteMarkedStyles⁽²⁸⁴⁾;
- DeleteUnusedStyles⁽²⁸⁶⁾.

6.1.1.3.103 TRichView.Reformat

Reformats the document.

Only formatted documents can be displayed or edited.

procedure Reformat;

(introduced in v1.9)

This method works like Format⁽²⁸⁸⁾, but keeps the selection (and caret position in TRichViewEdit⁽⁴⁶¹⁾).

Document must be formatted before calling this method.

This method is useful, for example, if you want to reformat document after changing margins.

6.1.1.3.104 TRichView.RefreshListMarkers

Recalculates list markers⁽¹⁷⁴⁾

procedure RefreshListMarkers;

(introduced in v1.8)

Call it (before calling Format⁽²⁸⁸⁾) if you changed ListType⁽⁷⁵⁵⁾ or FormatString⁽⁷⁵²⁾ of list style level of marker that is already inserted in TRichView. It's usually not necessary to use this method.

6.1.1.3.105 TRichView.RefreshListSequences

Recalculates numbered sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, and sidenotes⁽¹⁸¹⁾.

procedure RefreshSequences;

(introduced in v13)

Call it (before calling Format⁽²⁸⁸⁾) if you changed FootnoteNumbering⁽⁶³⁷⁾, EndnoteNumbering⁽⁶³⁷⁾ or SidenoteNumbering⁽⁶³⁷⁾ of TRVStyle⁽⁶³⁰⁾ component linked⁽²⁵³⁾ to this TRichView control, or a property of a numbered sequence item that was already inserted, to update existing document containing footnotes or endnotes.

6.1.1.3.106 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⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.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` ⁽²⁴⁰⁾.

See also methods of TRichView:

- `GetItemCheckpoint` ⁽²⁹⁸⁾;
- `SetCheckpointInfo` ⁽³⁵³⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also methods of RichViewEdit:

- `RemoveCheckpointEd` ⁽⁵⁴¹⁾;
- `RemoveCurrentCheckpoint` ⁽⁵⁴¹⁾.

See also:

- "Checkpoints" ⁽⁸⁷⁾;
- Modifying RichView items ⁽¹¹²⁾.

6.1.1.3.107 TRichView.ResetAnimation

Rewinds all inserted animated pictures to the first frame.

procedure `ResetAnimation`;

(introduced in version 10)

See also methods:

- `StartAnimation` ⁽³⁶⁶⁾;
- `StopAnimation` ⁽³⁶⁷⁾.

See also properties:

- `AnimationMode` ⁽²²²⁾.

See also:

- Animation in TRichView ⁽¹¹⁹⁾;
- Pictures ⁽¹⁶³⁾;
- *Hot-Pictures* ⁽¹⁶⁶⁾.

6.1.1.3.108 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` ⁽¹⁰³²⁾;
`SelectionOnly: Boolean): Boolean`;

(introduced in version 15; changed in version 18)

This method is available for Delphi 5+, C++Builder 2006+

The method uses `rvtfSaveDocParameters` and `rvtfSaveHeaderFooter` options from `RTFOptions` ⁽²⁴⁵⁾ property.

By default, inserted controls are not saved. You can save them yourself using `OnSaveComponentToFile` ⁽³⁹⁷⁾ event.

By default, items tags⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in OnWriteObjectProperties⁽⁴¹³⁾ event.

You can:

- completely change DocX code for certain items using OnSaveItemToFile⁽⁴⁰⁴⁾ event;
- insert OOXML codes⁽⁷⁰⁴⁾ in text;
- insert additional code in DocX in OnSaveDocXExtra⁽³⁹⁹⁾ event.

Return value: "successful saving?"

See also methods:

- SaveDocXToStream⁽³³⁴⁾;
- LoadDocX⁽³¹⁸⁾

See also properties:

- RTFOptions⁽²⁴⁵⁾;
- UseStyleTemplates⁽²⁵⁶⁾.

See also events:

- OnWriteHyperlink⁽⁴¹¹⁾;
- OnWriteObjectProperties⁽⁴¹³⁾;
- OnSaveComponentToFile⁽³⁹⁷⁾;
- OnSaveDocXExtra⁽³⁹⁹⁾;
- OnSaveItemToFile⁽⁴⁰⁴⁾;
- OnProgress⁽³⁸⁵⁾.

See also global variables and constants:

- RichViewSaveHyperlinksInDocXAsFields⁽¹⁰⁴⁸⁾

See also:

- Saving and loading in TRichView⁽¹²²⁾.

6.1.1.3.109 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⁽³³³⁾, but saves to a stream instead of a file.

See also

- LoadDocXFromStream⁽³¹⁹⁾

6.1.1.3.110 TRichView.SaveHTML

Exports TRichView document to HTML (or XHTML) file

```
function SaveHTML(const FileName: TRVUnicodeString1032;  
  Part: TRVHTMLSavingPart1012 = 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:

- HTMLSaveProperties²³⁷
- DocParameters²³⁰.Title⁴¹⁹, Author⁴¹⁶, and Comments⁴¹⁷.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile³⁹⁷ event.

By default, items tags⁹¹ are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink⁴¹¹ event.

You can save additional information about non-text objects in OnWriteObjectProperties⁴¹³ event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile⁴⁰⁴ event;
- insert additional code in HTML in OnSaveHTMLExtra⁴⁰¹ and OnSaveParaToHTML⁴⁰⁶ events;
- change how images are saved using OnHTMLSaveImage³⁷⁵ or OnSaveImage2⁴⁰² event,
- insert HTML codes⁷⁰⁴ in text.

Return value: "successful saving?"

See also methods:

- SaveHTMLToStream³³⁸

See also events:

- OnWriteHyperlink⁴¹¹;
- OnWriteObjectProperties⁴¹³;
- OnSaveComponentToFile³⁹⁷;
- OnSaveHTMLExtra⁴⁰¹;
- OnHTMLSaveImage³⁷⁵;
- OnSaveImage2⁴⁰²;
- OnSaveItemToFile⁴⁰⁴;
- OnSaveParaToHTML⁴⁰⁶;
- OnProgress³⁸⁵.

See also:

- Export to HTML ⁽¹²⁷⁾;
- RichViewSavePinHTML ⁽¹⁰⁴⁰⁾ typed constant;
- TFontInfo.Options ⁽⁷¹²⁾ (HTML codes);
- Saving and loading ⁽¹²²⁾.

6.1.1.3.111 TRichView.SaveHTML (deprecated)

Deprecated, use the new version of SaveHTML ⁽³³⁵⁾.

Exports TRichView document to HTML or XHTML file, using HTML tags like , , <div>, etc.

```
function SaveHTML(const FileName, Title,
  ImagesPrefix: TRVUnicodeString (1032);
  Options: TRVSaveOptions (1020)): 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 ⁽³³⁷⁾ 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 ⁽¹⁰²⁰⁾ for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile ⁽³⁹⁷⁾ event.

By default, items tags ⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink ⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in OnWriteObjectProperties ⁽⁴¹³⁾ event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile ⁽⁴⁰⁴⁾ event;
- insert additional code in HTML in OnSaveHTMLExtra ⁽⁴⁰¹⁾ and OnSaveParaToHTML ⁽⁴⁰⁶⁾ events;
- change how images are saved using OnHTMLSaveImage ⁽³⁷⁵⁾ or OnSaveImage2 ⁽⁴⁰²⁾ event,
- insert HTML codes ⁽⁷⁰⁴⁾ in text.

Return value: "successful saving?"

See also methods:

- SaveHTMLEx ⁽³³⁷⁾;
- SaveHTMLToStream ⁽³⁴⁰⁾.

See also events:

- OnWriteHyperlink ⁽⁴¹¹⁾;

- OnWriteObjectProperties⁽⁴¹³⁾;
- OnSaveComponentToFile⁽³⁹⁷⁾;
- OnSaveHTMLExtra⁽⁴⁰¹⁾;
- OnHTMLSaveImage⁽³⁷⁵⁾;
- OnSaveImage2⁽⁴⁰²⁾;
- OnSaveItemToFile⁽⁴⁰⁴⁾;
- OnSaveParaToHTML⁽⁴⁰⁶⁾;
- OnProgress⁽³⁸⁵⁾.

See also:

- Export to HTML⁽¹²⁷⁾;
- RichViewSavePinHTML⁽¹⁰⁴⁰⁾ typed constant;
- TFontInfo.Options⁽⁷¹²⁾ (HTML codes);
- Saving and loading⁽¹²²⁾.

6.1.1.3.112 TRichView.SaveHTMLEx (deprecated)

Deprecated, use SaveHTML⁽³³⁵⁾

Exports TRichView document to HTML or XHTML file, using CSS (Cascading Style Sheets)

```
function SaveHTMLEx(const FileName, Title, ImagesPrefix, ExtraStyles,
    ExternalCSS, CPPrefix: TRVUnicodeString(1032);
    Options: TRVSaveOptions(1020)): 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⁽⁶⁶²⁾) instead of saving CSS into HTML file).

CPPrefix — first part of *checkpoints*⁽⁸⁷⁾ 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⁽¹⁰²⁰⁾ for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile⁽³⁹⁷⁾ event.

By default, items *tags*⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in OnWriteObjectProperties⁽⁴¹³⁾ event.

You can:

- completely change HTML code for certain items using `OnSaveItemToFile` ⁽⁴⁰⁴⁾ event;
- insert additional code in HTML in `OnSaveHTMLExtra` ⁽⁴⁰¹⁾ and `OnSaveParaToHTML` ⁽⁴⁰⁶⁾ events;
- change how images are saved using `OnHTMLSaveImage` ⁽³⁷⁵⁾ or `OnSaveImage2` ⁽⁴⁰²⁾ event,
- insert HTML codes ⁽⁷⁰⁴⁾ 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` ⁽⁷⁰⁰⁾ 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` ⁽³³⁶⁾;
- `SaveHTMLToStreamEx` ⁽³⁴⁰⁾.

See also events:

- `OnWriteHyperlink` ⁽⁴¹¹⁾;
- `OnWriteObjectProperties` ⁽⁴¹³⁾;
- `OnSaveComponentToFile` ⁽³⁹⁷⁾;
- `OnSaveHTMLExtra` ⁽⁴⁰¹⁾;
- `OnHTMLSaveImage` ⁽³⁷⁵⁾;
- `OnSaveImage2` ⁽⁴⁰²⁾;
- `OnSaveItemToFile` ⁽⁴⁰⁴⁾;
- `OnSaveParaToHTML` ⁽⁴⁰⁶⁾;
- `OnProgress` ⁽³⁸⁵⁾.

See also:

- `Export to HTML` ⁽¹²⁷⁾;
- `TFontInfo.Options` ⁽⁷¹²⁾ (HTML codes);
- `RichViewSaveDivInHTMLEx` ⁽¹⁰⁴⁰⁾ typed constant;
- `Saving and loading` ⁽¹²²⁾.

6.1.1.3.113 TRichView.SaveHTMLToStream

Exports TRichView document to HTML (or XHTML) stream.

```
function SaveHTMLToStream(Stream: TStream;
  const Path: TRVUnicodeString(1032) = '';
  Part: TRVHTMLSavingPart(1012) = 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⁽²³⁷⁾
- DocParameters⁽²³⁰⁾.Title⁽⁴¹⁹⁾, Author⁽⁴¹⁶⁾, and Comments⁽⁴¹⁷⁾.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile⁽³⁹⁷⁾ event.

By default, items tags⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in OnWriteObjectProperties⁽⁴¹³⁾ event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile⁽⁴⁰⁴⁾ event;
- insert additional code in HTML in OnSaveHTMLExtra⁽⁴⁰¹⁾ and OnSaveParaToHTML⁽⁴⁰⁶⁾ events;
- change how images are saved using OnHTMLSaveImage⁽³⁷⁵⁾ or OnSaveImage2⁽⁴⁰²⁾ event,
- insert HTML codes⁽⁷⁰⁴⁾ in text.

Return value: "successful saving?"

See also methods:

- SaveHTML⁽³³⁵⁾

See also events:

- OnWriteHyperlink⁽⁴¹¹⁾;
- OnWriteObjectProperties⁽⁴¹³⁾;
- OnSaveComponentToFile⁽³⁹⁷⁾;
- OnSaveHTMLExtra⁽⁴⁰¹⁾;
- OnHTMLSaveImage⁽³⁷⁵⁾;
- OnSaveImage2⁽⁴⁰²⁾;
- OnSaveItemToFile⁽⁴⁰⁴⁾;
- OnSaveParaToHTML⁽⁴⁰⁶⁾;
- OnProgress⁽³⁸⁵⁾.

See also:

- Export to HTML⁽¹²⁷⁾;
- RichViewSavePinHTML⁽¹⁰⁴⁰⁾ typed constant;
- TFontInfo.Options⁽⁷¹²⁾ (HTML codes);
- Saving and loading⁽¹²²⁾.

6.1.1.3.114 TRichView.SaveHTMLToStream (deprecated)

Deprecated, use the new version of SaveHTMLToStream⁽³³⁸⁾.

Exports TRichView document as HTML or XHTML to the stream **Stream**, using HTML tags like , , <div>, etc.

```
function SaveHTMLToStream(Stream: TStream;
  const Path, Title, ImagesPrefix: TRVUnicodeString(1032);
  Options: TRVSaveOptions(1020)): Boolean; overload;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTML⁽³³⁶⁾, 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⁽³³⁶⁾.

6.1.1.3.115 TRichView.SaveHTMLToStreamEx (deprecated)

Deprecated, use SaveHTMLToStream⁽³³⁸⁾.

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(1032);
  Options: TRVSaveOptions(1020)): Boolean;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTMLEx⁽³³⁷⁾, 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⁽³³⁷⁾.

6.1.1.3.116 TRichView.SaveMarkdown

Saves the document (or the selected part, if **SelectionOnly=True**) to the file **FileName** as **Markdown**.

```
function SaveMarkdown(const FileName: TRVUnicodeString(1032);
  SelectionOnly: Boolean;
  CodePage: TRVCodePage(996) = CP_UTF8): Boolean;
```

(introduced in version 19, changed in version 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](#) ⁽²³⁹⁾.

See also methods:

- [SaveMarkdownToStream](#) ⁽³⁴¹⁾.

See also:

- [Saving and loading TRichView document](#) ⁽¹²²⁾.

6.1.1.3.117 TRichView.SaveMarkdownToStream

Saves the document (or the selected part, if **SelectionOnly=True**) to the **Stream** as **Markdown**.

```
function SaveMarkdownToStream(Stream: TStream;  
  const Path: TRVUnicodeString(1032); SelectionOnly: Boolean;  
  CodePage: TRVCodePage(996) = 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](#) ⁽²³⁹⁾.

See also methods:

- [SaveMarkdown](#) ⁽³⁴⁰⁾.

See also:

- [Saving and loading TRichView document](#) ⁽¹²²⁾.

6.1.1.3.118 TRichView.SavePicture

Saves a picture. This method is used when saving HTML, Markdown, and XML (using RichViewXML component) files.

```
function SavePicture(DocumentSaveFormat: TRVSaveFormat(1020);
  const ImagePrefix, Path: TRVUnicodeString(1032); var ImageSaveNo: Integer;
  OverrideImages, InlineImage: Boolean; BackgroundColor: TRVColor(936);
  gr: TRVGraphic(970); IsBackgroundImage = False): TRVUnicodeString(1032); virtual;
```

(changed in version 18, 19, 21)

Parameters:

gr – the image to save.

DocumentSaveFormat – output document format, see TRVSaveFormat⁽¹⁰²⁰⁾ 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⁽¹⁰⁷⁰⁾.

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⁽³⁷⁵⁾,
- OnSaveImage2⁽⁴⁰²⁾,
- OnSaveComponentToFile⁽³⁹⁷⁾,
- OnSaveItemToFile⁽⁴⁰⁴⁾ (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*⁽²³⁹⁾.*ImagesPrefix*⁽⁴⁴⁵⁾, **OverrideImages** = *rvmdsoOverrideImages* in *MarkdownProperties*⁽²³⁹⁾.*SaveOptions*⁽⁴⁴⁸⁾.

See also:

- [Export to HTML](#)⁽¹²⁷⁾.

6.1.1.3.119 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: TRVUnicodeString1032;  
    SelectionOnly: Boolean): Boolean;
```

(introduced in version 1.3; changed in version 18)

Options for saving are in [RTFOptions](#)⁽²⁴⁵⁾ property.

By default, inserted controls are not saved. You can save them yourself using [OnSaveComponentToFile](#)⁽³⁹⁷⁾ event.

By default, items tags⁽⁹¹⁾ are saved as targets of hypertext links. You can customize saving using [OnWriteHyperlink](#)⁽⁴¹¹⁾ event.

You can save additional information about non-text objects in [OnWriteObjectProperties](#)⁽⁴¹³⁾ event.

You can:

- completely change RTF code for certain items using [OnSaveItemToFile](#)⁽⁴⁰⁴⁾ event;
- insert additional code in RTF in [OnSaveRTFExtra](#)⁽⁴⁰⁶⁾ event;
- insert RTF codes⁽⁷⁰⁴⁾ in text.

Return value: "successful saving?"

See also methods:

- [SaveRTFToStream](#)⁽³⁴⁴⁾;
- [CopyRTF](#)⁽²⁸²⁾;
- [LoadRTF](#)⁽³²⁶⁾.

See also properties:

- [RTFOptions](#)⁽²⁴⁵⁾;
- [UseStyleTemplates](#)⁽²⁵⁶⁾.

See also events:

- [OnWriteHyperlink](#)⁽⁴¹¹⁾;
- [OnWriteObjectProperties](#)⁽⁴¹³⁾;
- [OnSaveComponentToFile](#)⁽³⁹⁷⁾;
- [OnSaveRTFExtra](#)⁽⁴⁰⁶⁾;
- [OnSaveItemToFile](#)⁽⁴⁰⁴⁾;
- [OnProgress](#)⁽³⁸⁵⁾.

See also:

- [Saving and loading in RichView](#)⁽¹²²⁾.

6.1.1.3.120 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⁽³⁴³⁾, but saves to a stream instead of a file.

6.1.1.3.121 TRichView.SaveRVF

Saves the document (or the selected part, if **SelectionOnly=True**) to the file **FileName** as RVF (RichView Format⁽¹²⁴⁾)

```
function SaveRVF(const FileName: TRVUnicodeString(1032);  
    SelectionOnly: Boolean): Boolean;
```

(changed in version 18)

Setting for RVF saving can be changed in the TRichView component editor⁽²¹⁸⁾.

Return value: "successful saving?"

See also properties:

- RVFOptions⁽²⁴⁷⁾.

See also methods:

- SaveRVFToStream⁽³⁴⁴⁾;
- LoadRVF⁽³²⁸⁾.

See also:

- Saving and loading RichView document⁽¹²²⁾;
- RVF⁽¹²⁴⁾.

6.1.1.3.122 TRichView.SaveRVFToStream

Saves the document (or the selected part, if **SelectionOnly=True**) to the **Stream** as RVF (RichView Format⁽¹²⁴⁾).

```
function SaveRVFToStream(Stream: TStream;  
    SelectionOnly: Boolean): Boolean;
```

Setting for RVF saving can be changed in the TRichView component editor⁽²¹⁸⁾.

Return value: "successful saving?"

See also properties:

- RVFOptions⁽²⁴⁷⁾.

See also methods:

- SaveRVF⁽³⁴⁴⁾;
- LoadRVFFromStream⁽³²⁹⁾;
- InsertRVFFromStream⁽³¹⁶⁾;
- AppendRVFFromStream⁽²⁷⁶⁾.

See also:

- Saving and loading RichView document ⁽¹²²⁾;
- RVF ⁽¹²⁴⁾.

6.1.1.3.123 TRichView.SaveText -W

The methods export document to ANSI (**SaveText**) or Unicode (**SaveTextW**) text file.

```
function SaveText(const FileName: TRVUnicodeString1032;
  LineWidth: Integer; CodePage: Cardinal=CP_ACP): Boolean;
function SaveTextW(const FileName: TRVUnicodeString1032;
  LineWidth: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName – output file name.

LineWidth is used for saving *breaks* ⁽¹⁶⁷⁾ (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** ⁽²⁵³⁾.DefCodePage ⁽⁶³⁵⁾ is used instead.

Return value: "successful saving?"

See also methods:

- SaveTextToStream, SaveTextToStreamW ⁽³⁴⁵⁾;
- LoadText, LoadTextW ⁽³³⁰⁾.

See also events:

- OnSaveImage2 ⁽⁴⁰²⁾;
- OnSaveComponentToFile ⁽³⁹⁷⁾;
- OnSaveItemToFile ⁽⁴⁰⁴⁾.

See also:

- Saving and loading RichView documents ⁽¹²²⁾.

6.1.1.3.124 TRichView.SaveTextToStream -W

The methods export document or selection as ANSI (**SaveTextToStream**) or Unicode (**SaveTextToStreamW**) text.

```
function SaveTextToStream(const Path: TRVUnicodeString1032;
  Stream: TStream; LineWidth: Integer;
  SelectionOnly, TextOnly: Boolean;
  CodePage: Cardinal=CP_ACP): Boolean;
function SaveTextToStreamW(const Path: TRVUnicodeString1032;
  Stream: TStream; LineWidth: Integer;
  SelectionOnly, TextOnly: Boolean): Boolean;
```

(introduced in version 1.3; changed in version 18)

Parameters

LineWidth is used for saving *breaks* ⁽¹⁶⁷⁾ (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` ⁽⁴⁰²⁾, `OnSaveComponentToFile` ⁽³⁹⁷⁾, and `OnSaveItemToFile` ⁽⁴⁰⁴⁾ 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` ⁽²⁵³⁾.`DefCodePage` ⁽⁶³⁵⁾ is used instead.

Return value: "successful saving?"

See also methods:

- `SaveText`, `SaveTextW` ⁽³⁴⁵⁾;
- `LoadTextFromStream`, `LoadTextFromStreamW` ⁽³³¹⁾.

See also events:

- `OnSaveImage2` ⁽⁴⁰²⁾;
- `OnSaveComponentToFile` ⁽³⁹⁷⁾;
- `OnSaveItemToFile` ⁽⁴⁰⁴⁾.

See also:

- Saving and loading RichView documents ⁽¹²²⁾.

6.1.1.3.125 TRichView.SearchText -A -W

The methods search for the substring *s* in the document.

```
function SearchText(s: String;
  SrchOptions: TRVSearchOptions(1024)): Boolean;
function SearchTextA(s: TRVAnsiString(993);
  SrchOptions: TRVSearchOptions(1024)): Boolean;
function SearchTextW(s: TRVUnicodeString(1032);
  SrchOptions: TRVSearchOptions(1024)): Boolean;
```

When found, these methods select substring and scroll to it (this behavior can be modified in `OnTextFound` ⁽⁴¹⁰⁾ 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).

`rsvroSmartStart` is used only if `rsvroFromStart` is not included. If `rsvroSmartStart` 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 `rsvroFromStored` is included in **SrchOptions**, the last stored search position⁽³⁶⁷⁾ if used instead of selection.

If `rsvroMultitem` is included, the search can match substrings of several text items⁽¹⁶¹⁾. If not included, the text is searched in each text item separately. For example, if the document contains the text **Hello**, 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`⁽²⁵³⁾.`DefCodePage`⁽⁶³⁵⁾) 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⁽¹⁵⁴⁾

Note: There is incompatibility with programs written with the older version of TRichView (1.0 – 1.1.4). Now if `rsvroDown` is not included in `SrchOptions`, the methods search upwards. In the older versions there was no `rsvroDown` option, and `SearchText` searched downwards only.

When the text is found, `OnTextFound`⁽⁴¹⁰⁾ 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`⁽⁴¹⁰⁾ 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`⁽⁵⁴³⁾

See also global functions:

- `GetRVSearchOptions`⁽⁹⁸⁷⁾.

See also:

- Searching in TRichView and TRichViewEdit⁽¹¹⁰⁾.

6.1.1.3.126 TRichView.SelectAll

Selects the whole document for copying to the Clipboard

procedure `SelectAll`;

This method does not repaint document, call `Invalidate`.

In `RichViewEdit`⁽⁴⁶¹⁾ this method moves the caret to the end of document. The method generates `OnSelect`⁽⁴⁰⁸⁾ event.

This method must be called only when the document is formatted.

See also methods:

- `SetSelectionBounds`⁽³⁶⁵⁾;
- `Deselect`⁽²⁸⁶⁾.

See also:

- Selecting in RichView document⁽¹⁰⁷⁾.

See also:

- `Edit`⁽⁹⁵⁹⁾ method of table cell.

6.1.1.3.127 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⁽¹⁶⁸⁾ containing the control **actrl**.

This method repaints document.

The method generates `OnSelect`⁽⁴⁰⁸⁾ event.

In `TRichViewEdit`⁽⁴⁶¹⁾, if this control's item is resizable (see `rvepResizable` extra-property⁽¹⁰⁰⁰⁾) and resizing with mouse is not forbidden (see `rvoNoImageResize` in `EditorOptions`⁽⁴⁷⁵⁾), 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`⁽³⁶⁵⁾;
- `Deselect`⁽²⁸⁶⁾.

See also:

Selecting in RichView document⁽¹⁰⁷⁾.

6.1.1.3.128 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⁽⁴⁰⁸⁾ 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⁽¹⁰⁷⁾.

6.1.1.3.129 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(998));
```

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters⁽²²⁸⁾ property.

This method repaints document.

In RichViewEdit⁽⁴⁶¹⁾ this method moves the caret to the end of the selected word.

The method generates OnSelect⁽⁴⁰⁸⁾ 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 *rvoDbClickSelectsWord* is in Options⁽²⁴⁰⁾.

See also events:

- OnRVRightClick⁽³⁹⁷⁾;
- OnRVDbClick⁽³⁹¹⁾.

See also properties:

- Delimiters⁽²²⁸⁾.

See also methods:

- SelectAll⁽³⁴⁸⁾;
- GetWordAt⁽³¹⁴⁾;
- SetSelectionBounds⁽³⁶⁵⁾;
- GetSelText⁽³¹³⁾.

See also methods of TRichViewEdit:

- `SelectCurrentWord`⁵⁴⁵.

See also:

- `Selecting in RichView document`¹⁰⁷.

6.1.1.3.130 TRichView.SetAddParagraphMode

Sets mode for adding items from new line for all `Add***` methods¹⁰².

```
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`¹⁶⁷ or `tables`¹⁷³) 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 documents¹³⁸

Example 1:

```
with MyRichView do
begin
  Clear279;
  SetAddParagraphMode(True); // default mode
  AddNL270('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
end;
```

```

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 ⁽⁹⁵⁾;
- Building a document ⁽¹⁰²⁾.

6.1.1.3.131 TRichView.SetBreakInfo

Changes main properties of the item of *break* ⁽¹⁶⁷⁾ (horizontal line) type.

```

procedure SetBreakInfo( ItemNo: Integer; AWidth: TRVStyleLength (1027);
  AStyle: TRVBreakStyle (995); AColor: TRVColor (996); const ATag: TRVTag (1029) );

```

Parameters:

ItemNo – index of the item. The item must be of *break* ⁽¹⁶⁷⁾ type (*rvsBreak* ⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError* ⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount* ⁽²³⁷⁾ -1, *GetItemStyle* ⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells ⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData* ⁽⁹⁰⁷⁾.*SetBreakInfo*.

AWidth – line width (or rectangle height). This value is measured in *Style* ⁽²⁵³⁾.*Units* ⁽⁶⁵⁵⁾.

AStyle – visual style of this *break*, see *TRVBreakStyle* ⁽⁹⁹⁵⁾ for possible values.

AColor – line color. If it is equal to *rvclNone* ⁽¹⁰³⁸⁾, *Style* ⁽²⁵³⁾.*TextStyles* ⁽⁶⁵⁴⁾ [0].*Color* ⁽⁷⁰¹⁾ is used.

ATag – *tag* of the item. You can use value returned by *GetBreakInfo* ⁽²⁸⁹⁾ or *GetItemTag* ⁽³⁰²⁾ for this item.

Instead of this method, you can use *SetItemExtraIntPropertyEx* ⁽³⁵⁸⁾ and *SetItemTag* ⁽³⁶⁰⁾ 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* ⁽³⁵⁸⁾ and *SetItemExtraStrProperty* ⁽³⁵⁹⁾.

Example

```

RichView2.SetBreakInfo( ItemNo, 1, rvbsLine, clNone, 'Break Tag' );

```

See also methods:

- *GetBreakInfo* ⁽²⁸⁹⁾;
- *GetItemStyle* ⁽³⁰²⁾;
- *SetItemExtraIntProperty* ⁽³⁵⁸⁾;
- *SetItemExtraStrProperty* ⁽³⁵⁹⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also methods of RichViewEdit:

- `SetBreakInfoEd`⁽⁵⁴⁷⁾;
- `SetCurrentBreakInfo`⁽⁵⁵¹⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `Tags`⁽⁹¹⁾.

6.1.1.3.132 TRichView.SetBulletInfo

Changes main properties of the item of *bullet*⁽¹⁷⁰⁾ (image from ImageList) type.

VCL and LCL:

```
procedure SetBulletInfo(ItemNo: Integer; const AName: TRVUnicodeString(1032);
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag(1029));
```

(changed in version 18)

FireMonkey:

```
procedure SetBulletInfo(ItemNo: Integer; const AName: TRVUnicodeString(1032);
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag(1029);
  AImageWidth, AImageHeight: TRVStyleLength(1027));
```

Parameters:

ItemNo – index of the item. The item must be of *bullet*⁽¹⁷⁰⁾ or *hotspot*⁽¹⁷²⁾ type (*rvsBullet* or *rvsHotspot*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`SetBulletInfo`.

AName – name of *bullet*⁽¹⁷⁰⁾, any string without line break (CR, LF) characters. It can also be set using `SetItemText`⁽³⁶⁰⁾ method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using `SetItemExtraIntPropertyEx`⁽³⁵⁸⁾ method.

ATag – *tag* of the item. You can use value returned by `GetBulletInfo`⁽²⁸⁹⁾ or `GetItemTag`⁽³⁰²⁾ for this item. The *tag* can also be set by `SetItemTag`⁽³⁶⁰⁾ 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`⁽³⁵⁸⁾ and `SetItemExtraStrProperty`⁽³⁵⁹⁾.

Example

```
RichView1.SetBulletInfo(ItemNo, 'my image', 0, nil, 'bullet tag');
```

See also methods:

- `GetBulletInfo`⁽²⁸⁹⁾;
- `GetItemStyle`⁽³⁰²⁾;
- `SetItemExtraIntProperty`⁽³⁵⁸⁾;
- `SetItemExtraStrProperty`⁽³⁵⁹⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also methods of RichViewEdit:

- `SetBulletInfoEd`⁽⁵⁴⁸⁾;
- `SetCurrentBulletInfo`⁽⁵⁵²⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `Tags`⁽⁹¹⁾.

6.1.1.3.133 TRichView.SetCheckpointInfo

Creates a new *checkpoint* or modifies the existing *checkpoint* associated with the **ItemNo**-th item

```
procedure SetCheckpointInfo(ItemNo: Integer; const ATag: TRVTag(1029);
const AName: TRVUnicodeString(1032); ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to `ItemCount`⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData`⁽⁹⁰⁷⁾.`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`⁽³⁷²⁾ 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`⁽²⁴⁰⁾).

See also methods:

- `GetItemCheckpoint`⁽²⁹⁸⁾;

- RemoveCheckpoint⁽³³²⁾.

See also properties:

- ItemCount⁽²³⁷⁾.

See also methods of RichViewEdit:

- SetCheckpointInfoEd⁽⁵⁴⁹⁾;
- SetCurrentCheckpointInfo⁽⁵⁵³⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- Checkpoints⁽⁸⁷⁾;
- Tags⁽⁹¹⁾.

6.1.1.3.134 TRichView.SetControlInfo

Changes main properties of the item of control⁽¹⁶⁸⁾ type.

```
function SetControlInfo(ItemNo: Integer; const AName: TRVUnicodeString(1032);
  AVAlign: TRVVAlign(1033); const ATag: TRVTag(1029)): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control⁽¹⁶⁸⁾ type (*rvsComponent*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount*⁽²³⁷⁾-1, *GetItemStyle*⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData*⁽⁹⁰⁷⁾.*SetControlInfo*.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using *SetItemText*⁽³⁶⁰⁾ method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see *TRVVAlign*⁽¹⁰³³⁾ for possible options.

ATag – *tag* of the item. Do not confuse with *Tag* property of the control! You can use value returned by *GetControlInfo*⁽²⁹³⁾ or *GetItemTag*⁽³⁰²⁾ for this item. The *tag* can also be set by *SetItemTag*⁽³⁶⁰⁾ 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*⁽³⁵⁸⁾ and *SetItemExtraStrProperty*⁽³⁵⁹⁾.

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⁽²⁹³⁾;
- GetItemStyle⁽³⁰²⁾;

- `Format`⁽²⁸⁸⁾;
- `SetItemExtraIntProperty`⁽³⁵⁸⁾;
- `SetItemExtraStrProperty`⁽³⁵⁹⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also methods of RichViewEdit:

- `SetControlInfoEd`⁽⁵⁵⁰⁾;
- `SetCurrentControlInfo`⁽⁵⁵⁴⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `Tags`⁽⁹¹⁾.

6.1.1.3.135 TRichView.SetFooter

Assigns the specified document for using as a footer.

```
procedure SetFooter(Footer: TCustomRVData(957);  
HeaderType: TRVHFTType(1011) = rvhftNormal);
```

(introduced in version 17)

Parameters:

Footer – document which will be used as a footer (of the type specified in **HFTType**);

HFTType – 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`⁽⁷⁷⁸⁾ 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 `rvtfSaveHeaderFooter` is included in `RTFOptions`⁽²⁴⁵⁾;
- **Footer** will be saved after the main document in HTML, if `HTMLSaveProperties`⁽²³⁷⁾.`SaveHeaderAndFooter`⁽⁴³⁷⁾ = `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⁽²⁵³⁾ to each own `TRVStyle`⁽⁶³⁰⁾ component. Values of `UseStyleTemplates`⁽²⁵⁶⁾ must be the same for the main document and all headers and footers. `UseStyleTemplates`⁽²⁵⁶⁾ = `True`, `MainRVStyle`⁽⁶⁴⁷⁾ of `RVStyles` of headers and footers must be equal to `RVStyle` of the main document (to allow them using the same collections of `StyleTemplates`⁽⁶⁵²⁾).

Example:

```
RichView1.Clear(279);  
RichView2.Clear(279);
```

```

RichView1.RTFReadProperties.ReadDocParameters(457) := True;
RichView1.SetFooter(RichView2.RVData(247));
RichView1.LoadRTF(326)('test.rtf');
RichView1.Format(288); // contains document
RichView2.Format(288); // contains header
RVPrint1.AssignSource(771)(RichView1);
if RichView2.ItemCount(237)>0 then
  RVPrint1.SetFooter(778)(RichView2.RVData)
else
  RVPrint1.SetFooter(778)(nil);
RVPrint1.AssignDocParameters(771)(RichView1.DocParameters(230));
RVPrint1.FormatPages(774)(rvdoAll);
RVPrint1.Print(776)('Doc with footer', 1, False);

```

See also:

- SetHeader⁽³⁵⁶⁾.

See also properties of TRVStyle⁽⁶³⁰⁾:

- MainRVStyle⁽⁶⁴⁷⁾.

6.1.1.3.136 TRichView.SetHeader

Assigns the specified document for using as a header.

```

procedure SetHeader(Header: TCustomRVData(957);
  HeaderType: TRVHFTType(1011) = rvhftNormal);

```

(introduced in version 17)

Parameters:

Header – document which will be used as a header (of the type specified in **HFTType**);

HFTType – 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⁽⁷⁷⁹⁾ 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⁽²⁴⁵⁾;
- **Header** will be saved before the main document in HTML, if HTMLSaveProperties⁽²³⁷⁾.SaveHeaderAndFooter⁽⁴³⁷⁾ = 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⁽²⁵³⁾ to each own TRVStyle⁽⁶³⁰⁾ component. Values of UseStyleTemplates⁽²⁵⁶⁾ must be the same for the main document and all headers and footers. UseStyleTemplates⁽²⁵⁶⁾ = True, MainRVStyle⁽⁶⁴⁷⁾ of RVStyles of headers and footers must be equal to RVStyle of the main document (to allow them using the same collections of StyleTemplates⁽⁶⁵²⁾).

Example:

```

RichView1.Clear(279);
RichView2.Clear(279);
RichView1.RTFReadProperties.ReadDocParameters(457) := True;
RichView1.SetHeader(247)(RichView2.RVData);
RichView1.LoadRTF(326)('test.rtf');
RichView1.Format(288); // contains document
RichView2.Format(288); // contains header
RVPrint1.AssignSource(771)(RichView1);
if RichView2.ItemCount(237)>0 then
  RVPrint1.SetHeader(779)(RichView2.RVData)
else
  RVPrint1.SetHeader(779)(nil);
RVPrint1.AssignDocParameters(771)(RichView1.DocParameters(230));
RVPrint1.FormatPages(774)(rvdoAll);
RVPrint1.Print(776)('Doc with header', 1, False);

```

See also:

- SetFooter⁽³⁵⁶⁾.

See also properties of TRVStyle⁽⁶³⁰⁾:

- MainRVStyle⁽⁶⁴⁷⁾.

6.1.1.3.137 TRichView.SetHotspotInfo

Changes main properties of the item of *hotspot*⁽¹⁷²⁾ (image from ImageList, hyperlink) type.

VCL and LCL:

```

procedure SetHotspotInfo(ItemNo: Integer;
  const AName: TRVUnicodeString(1032);
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList;
  const ATag: TRVTag(1029));

```

(changed in version 18)

FireMonkey:

```

procedure SetHotspotInfo(ItemNo: Integer;
  const AName: TRVUnicodeString(1032);
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList;
  const ATag: TRVTag(1029);
  AImageWidth, AImageHeight: TRVStyleLength(1027));

```

Parameters:

ItemNo – index of the item. The item must be of *hotspot*⁽¹⁷²⁾ type (*rvsHotspot*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount*⁽²³⁷⁾-1, *GetItemStyle*⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use *Cell.GetRVData*⁽⁹⁰⁷⁾.SetHotspotInfo.

AName – name of *hotspot*⁽¹⁷²⁾, any string without line break (CR, LF) characters. It can also be set using `SetItemText`⁽³⁶⁰⁾ method.

AlmageList – not used, reserved, set it to *nil*..

AlmageIndex – index of image in image list. It can also be set using `SetItemExtraIntPropertyEx`⁽³⁵⁸⁾ method.

AlmageIndex – 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`⁽³⁵⁸⁾ method.

ATag – *tag* of the item. You can use value returned by `GetHotspotInfo`⁽²⁹⁵⁾ or `GetItemTag`⁽³⁰²⁾ for this item. The *tag* can also be set by `SetItemTag`⁽³⁶⁰⁾ method.

AlmageWidth, AlmageHeight – 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`⁽³⁵⁸⁾ and `SetItemExtraStrProperty`⁽³⁵⁹⁾.

Example

```
RichView1.SetHotspotInfo(ItemNo, 'my image', 0, 0, nil, 'hotspot tag');
```

See also methods:

- `GetHotspotInfo`⁽²⁹⁵⁾;
- `GetItemStyle`⁽³⁰²⁾;
- `SetItemExtraIntProperty`⁽³⁵⁸⁾;
- `SetItemExtraStrProperty`⁽³⁵⁹⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also methods of `RichViewEdit`:

- `SetHotspotInfoEd`⁽⁵⁶⁰⁾;
- `SetCurrentHotspotInfo`⁽⁵⁵⁵⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `Tags`⁽⁹¹⁾.

6.1.1.3.138 `TRichView.SetItemExtraIntProperty -Ex`

The methods change a value of the specified integer property of the **ItemNo**-th item.

```
function SetItemExtraIntProperty(ItemNo: Integer;  
Prop: TRVExtraItemProperty(1000); 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⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.SetItemExtraIntProperty[Ex]

In **SetItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty⁽¹⁰⁰⁰⁾. 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⁽¹⁰⁵¹⁾

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]⁽³⁰⁰⁾;
- SetItemExtraStrProperty⁽³⁵⁹⁾;
- Format⁽²⁸⁸⁾.

See also methods of TRichViewEdit

- SetItemExtraIntProperty[Ex]Ed⁽⁵⁶²⁾;
- SetCurrentItemExtraIntProperty[Ex]⁽⁵⁵⁶⁾.

6.1.1.3.139 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(1005);
  const Value: TRVUnicodeString(1032)): Boolean;
function SetItemExtraStrProperty(ItemNo, Prop: Integer;
  const Value: TRVUnicodeString(1032)): 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⁽¹⁰⁰⁵⁾. 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⁽¹⁰⁵⁶⁾

ItemNo – index of the item, from 0 to ItemCount⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.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]` ⁽³⁰⁰⁾;
- `SetItemExtraIntProperty[Ex]` ⁽³⁵⁸⁾.

See also methods of TRichViewEdit:

- `SetItemExtraStrProperty[Ex]Ed` ⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrProperty[Ex]` ⁽⁵⁵⁷⁾.

6.1.1.3.140 TRichView.SetItemTag

Changing *tag* of the **ItemNo**-th item

```
procedure SetItemTag(ItemNo: Integer; const ATag: TRVTag (1029));
```

Items are indexed from 0 to `ItemCount` ⁽²³⁷⁾-1. Items of subdocuments (table cells ⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData` ⁽⁹⁰⁷⁾.
`.SetItemTag`.

Method type:  viewer-style. Reformatting or repainting is not necessary.

Example

```
RichView1.SetItemTag(ItemNo, 'tag #1');
```

See also methods:

- `GetItemTag` ⁽³⁰²⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also methods of RichViewEdit:

- `SetItemTagEd` ⁽⁵⁶³⁾;
- `SetCurrentTag` ⁽⁵⁶⁰⁾.

See also:

- `Modifying RichView items` ⁽¹¹²⁾;
- `Tags` ⁽⁹¹⁾.

6.1.1.3.141 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 (993));
```

```
procedure SetItemTextW(ItemNo: Integer; const s: TRVUnicodeString (1032));
```

(Introduced in version 1.3, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.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⁽⁷⁰⁰⁾. If it is equal to DEFAULT_CHARSET, and for non-text items, Style⁽²⁵³⁾.DefCodePage⁽⁶³⁵⁾ is used.

SetItemText works:

- like **SetItemTextA**, in Delphi 2007 and older
- like **SetItemTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus⁽¹⁵⁴⁾

Methods type:  viewer-style. Reformatting and repainting are required only for text items.

See also methods:

- GetItemText -W -A⁽³⁰³⁾;
- Format⁽²⁸⁸⁾.

See also methods of TCustomRichViewEdit:

- SetItemTextEd -A -W⁽⁵⁶⁴⁾;
- SetCurrentItemText -A -W⁽⁵⁵⁷⁾.

See also:

- Modifying RichView Items⁽¹¹²⁾;
- Unicode in RichView⁽¹³⁰⁾.

6.1.1.3.142 TRichView.SetItemVAlign

Changing vertical alignment (position relative to the line) of the **ItemNo**-th item

```
procedure SetItemVAlign(ItemNo: Integer; VAlign: TRVVAlign(1033));
```

(introduced in version 12)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use Cell.GetRVData⁽⁹⁰⁷⁾.SetItemVAlign.

VAlign – new value of vertical alignment.

This method must not be called for text items⁽¹⁶¹⁾, tables⁽¹⁷³⁾, *breaks*⁽¹⁶⁷⁾, tabs⁽¹⁶²⁾, list markers⁽¹⁷⁴⁾, footnotes⁽¹⁸⁰⁾ and endnotes⁽¹⁷⁸⁾.

Method type:  viewer-style. Requires reformatting and repainting.

See also methods:

- `GetItemVAlign` ⁽³⁰³⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also methods of RichViewEdit:

- `SetItemVAlignEd` ⁽⁵⁶⁵⁾;
- `SetCurrentItemVAlign` ⁽⁵⁵⁸⁾.

See also:

- `Modifying RichView items` ⁽¹¹²⁾

6.1.1.3.143 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` ⁽²³⁷⁾-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 ⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVData` ⁽⁹⁰⁷⁾.`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` ⁽²³⁷⁾, 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` ⁽¹³⁸⁾).

AListNo – style of this list maker. This is an index in the collection of list styles (`Style` ⁽²⁵³⁾.`ListStyles` ⁽⁶⁴⁶⁾).

AListLevel – list level. This is an index in the collection of list levels (`Style.ListStyles[AListNo].Levels` ⁽⁷³²⁾).

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` ⁽²⁵³⁾.`ParaStyles` ⁽⁶⁴⁸⁾). 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 marked paragraph, call `Add` ⁽²⁷⁰⁾ to add empty text item after the list marker. See also: rules for valid documents ⁽¹³⁸⁾

Return value: item index of the marker.

See also methods:

- `GetListMarkerInfo`⁽³⁰⁷⁾;
- `Format`⁽²⁸⁸⁾.

See also properties:

- `Style`⁽²⁵³⁾;
- `ItemCount`⁽²³⁷⁾.

See also methods of TCustomRichViewEdit

- `ApplyListStyle`⁽⁴⁹⁰⁾;
- `RemoveLists`⁽⁵⁴²⁾;
- `ChangeListLevels`⁽⁵⁰⁰⁾.

See also:

- `List Markers`⁽¹⁷⁴⁾.

6.1.1.3.144 TRichView.SetPictureInfo

Changes main properties of the item of picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾ type.

```
function SetPictureInfo(ItemNo: Integer; const AName: TRVUnicodeString(1032);
  Agr: TRVGraphic(970); AAlign: TRVAlign(1033);
  const ATag: TRVTag(1029)): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾ type (*rvsPicture* or *rvsHotPicture*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item. Items of subdocuments (table cells⁽⁸⁹⁵⁾) are not included in the items range of the main document; for items in cells, use `Cell.GetRVDData`⁽⁹⁰⁷⁾.`SetPictureInfo`.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using `SetItemText`⁽³⁶⁰⁾ method.

Agr – graphic object. If this item does not use a shared image (by default), **Agr** must be a unique graphic object (you cannot assign the same graphic object to two or more items) or the value returned by `GetPictureInfo`⁽³¹⁰⁾ for this item (the image is shared if *rvepShared* extra item property⁽¹⁰⁰⁰⁾ is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by `GetPictureInfo`⁽³¹⁰⁾ for this item, the previously used graphic object is freed.

AAlign – vertical alignment of the picture.

ATag – *tag* of the item. You can use value returned by `GetPictureInfo`⁽³¹⁰⁾ or `GetItemTag`⁽³⁰²⁾ for this item. The *tag* can also be set by `SetItemTag`⁽³⁶⁰⁾ 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`³⁵⁸ and `SetItemExtraStrProperty`³⁵⁹ (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.SetItemExtraIntProperty358(ItemNo1, rvepShared, 1);
RichView1.SetPictureInfo(ItemNo2, '', bmp, rvvaBaseline, '');
RichView1.SetItemExtraIntProperty358(ItemNo2, rvepShared, 1);
// you must free bmp yourself when RichView1 is cleared
```

See also methods:

- `GetPictureInfo`³¹⁰;
- `GetItemStyle`³⁰²;
- `Format`²⁸⁸;
- `SetItemExtraIntProperty`³⁵⁸;
- `SetItemExtraStrProperty`³⁵⁹.

See also properties:

- `ItemCount`²³⁷.

See also methods of RichViewEdit:

- `SetPictureInfoEd`⁵⁶⁵;
- `SetCurrentPictureInfo`⁵⁵⁹.

See also:

- `Modifying RichView items`¹¹²;
- `Item types`⁸⁵;
- `Tags`⁹¹.

6.1.1.3.145 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⁽¹⁶¹⁾, 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⁽⁷⁰¹⁾>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⁽⁷⁰¹⁾>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⁽⁴⁶¹⁾, the caret is always at the end of the selection, so this method can be used to move caret.

Selection in table⁽¹⁷³⁾ cells

Before making selecting inside a table cell, activates its editing:

```
var RVData: TCustomRVFormattedData(957);
RVData := TCustomRVFormattedData(957) (Table.Cells(857) [r,c].Edit(959));
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⁽⁴⁰⁸⁾ event.

There are two useful methods to work with SetSelectionBounds: GetOffsBeforeItem⁽³⁰⁹⁾ and GetOffsAfterItem⁽³⁰⁸⁾.

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⁽⁹⁸⁴⁾ unit instead.

See also methods:

- SelectAll⁽³⁴⁸⁾;
- Deselect⁽²⁸⁶⁾;
- GetSelectionBounds⁽³¹²⁾.

See also events:

- OnSelect⁽⁴⁰⁸⁾.

See also:

- Working with selection⁽¹⁰⁷⁾;
- How to: move a caret to the beginning or to the end of document in TRichViewEdit⁽¹⁰⁶⁵⁾;
- Example how to move caret to the beginning of paragraph⁽⁵⁹²⁾.

See also:

- Edit⁽⁹⁵⁹⁾ method of table cell.

6.1.1.3.146 TRichView.StartAnimation

Starts animation of pictures.

```
procedure StartAnimation;
```

(introduced in version 10)

Alternatively, animation may start automatically when document is formatted⁽²⁸⁸⁾, if AnimationMode⁽²²²⁾ = *rvaniOnFormat*.

See also methods:

- ResetAnimation⁽³³³⁾;
- StopAnimation⁽³⁶⁷⁾.

See also properties:

- AnimationMode⁽²²²⁾.

See also:

- Animation in TRichView⁽¹¹⁹⁾;
- Pictures⁽¹⁶³⁾;
- *Hot-Pictures*⁽¹⁶⁶⁾.

6.1.1.3.147 TRichView.StartLiveSpelling

Starts live spelling check (runs a live spelling thread).

procedure StartLiveSpelling;

(introduced in v1.9)

A check starts only if:

- OnSpellingCheck⁽⁴⁰⁹⁾ event is assigned, or
- [in FireMonkey, in TRichViewEdit⁽⁴⁶¹⁾] CheckSpelling⁽⁴⁷²⁾ = *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⁽⁴⁷⁹⁾.

See also:

- Live spelling check in TRichView⁽¹³²⁾
- TRVSpellChecker⁽⁷⁸⁴⁾.StartLiveSpelling

6.1.1.3.148 TRichView.StopAnimation

Stops animation of pictures.

procedure StopAnimation;

(introduced in version 10)

Clear⁽²⁷⁹⁾ may stop animations as well.

See also methods:

- StartAnimation⁽³⁶⁶⁾;
- ResetAnimation⁽³³³⁾.

See also properties:

- AnimationMode⁽²²²⁾.

See also:

- Animation in TRichView⁽¹¹⁹⁾;
- Pictures⁽¹⁶³⁾;
- *Hot-Pictures*⁽¹⁶⁶⁾.

6.1.1.3.149 TRichView.StoreSearchResult

Stores the position in the document to allow searching from this place.

procedure StoreSearchResult(ARVData: TCustomRVData⁽⁹⁵⁷⁾;
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⁽⁴¹⁰⁾ event to store the position of the search result. Then you can continue searching from this position by calling TRichView.SearchText⁽³⁴⁶⁾ with *rvsoFromStored*⁽¹⁰²⁴⁾ in the Options parameter (or TRichViewEdit.SearchText⁽⁵⁴³⁾ with *rvseoFromStored*⁽⁹⁹⁹⁾ in the Options parameter

See also:

- Searching in TRichView and TRichViewEdit⁽¹¹⁰⁾.

6.1.1.3.150 TRichView.UpdatePalettelInfo

Call this method when screen color resolution changed (see WM_DISPLAYCHANGE window message). You need not to call this method if DoInPaletteMode⁽²³⁴⁾=rvpaDoNothing.

procedure UpdatePalettelInfo;

Example⁽²³⁵⁾

You also need to call UpdatePalettelInfo 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⁽²³⁴⁾.

See also methods of TRVPrint:

- UpdatePalettelInfo⁽⁸²⁸⁾.

6.1.1.4 Events

Derived from TCustomRichView

- OnAddStyle⁽³⁷⁰⁾
- OnAfterDrawImage⁽³⁷⁰⁾
- OnAssignImageFileName⁽³⁷¹⁾
- OnCheckpointVisible⁽³⁷²⁾
- OnCMHintShow⁽³⁷²⁾ [VCL,LCL]
- OnControlAction⁽³⁷³⁾
- OnCopy⁽³⁷⁴⁾
- OnGetItemCursor⁽³⁷⁴⁾
- OnGetSpellingSuggestions⁽³⁷⁵⁾ [FMX]
- OnHTMLSaveImage⁽³⁷⁵⁾
- OnImportFile⁽³⁷⁸⁾
- OnImportPicture⁽³⁷⁸⁾
- OnItemAction⁽³⁸⁰⁾
- OnItemHint⁽³⁸¹⁾
- OnJump⁽³⁸²⁾
- OnLoadCustomFormat⁽³⁸³⁾
- OnLoadDocument⁽³⁸⁴⁾
- OnNewDocument⁽³⁸⁴⁾

- OnPaint³⁸⁴
- OnProgress³⁸⁵
- OnReadField³⁸⁶
- OnReadHyperlink³⁸⁸
- OnReadMergeField³⁹⁰
- OnRVDbClick³⁹¹
- OnRVFControlNeeded³⁹²
- OnRVFImageListNeeded³⁹³
- OnRVFPictureNeeded³⁹³
- OnRVMouseDown³⁹⁴
- OnRVMouseMove³⁹⁵
- OnRVMouseUp³⁹⁶
- OnRVRightClick³⁹⁷
- OnSaveComponentToFile³⁹⁷
- OnSaveDocXExtra³⁹⁹
- OnSaveHTMLExtra⁴⁰¹
- OnSaveImage2⁴⁰²
- OnSaveItemToFile⁴⁰⁴
- OnSaveParaToHTML⁴⁰⁶
- OnSaveRTFExtra⁴⁰⁶
- OnSelect⁴⁰⁸
- OnSpellingCheck⁴⁰⁹
- OnStyleTemplatesChange⁴¹⁰
- OnTextFound⁴¹⁰
- OnWriteHyperlink⁴¹¹
- OnWriteObjectProperties⁴¹³

Derived from TRVScroller⁸¹³

- OnHScrolled⁸²¹
- OnVScrolled⁸²¹

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: TCustomRichView210;
  StyleInfo: TCustomRVInfo693) of object;
```

property OnAddStyle: TRVAddStyleEvent;

(introduced in version 11)

This event occurs when a new style is added (to TextStyles⁶⁵⁴, ParaStyles⁶⁴⁸, or ListStyles⁶⁴⁶ collections of the linked TRVStyle⁶³⁰ 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 used²⁵⁶) auto-correction of ParaStyleTemplateId⁷⁰⁵ property of text styles;
- (in editor) changing the current text style on keyboard language change (see *rvoAutoSwitchLang* in EditorOptions⁴⁷⁵);
- (in editor) ApplyStyleTemplate⁴⁹³, ApplyParaStyleTemplate⁴⁹¹, ApplyTextStyleTemplate⁴⁹⁴;
- (in editor, if style templates are used²⁵⁶) when a user presses **Enter** (because of applying NextId⁷³⁶ 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 pictures¹⁶³ and hot-pictures¹⁶⁶.

VCL and LCL:

type

```
TRVAfterDrawImageEvent = procedure (Sender: TCustomRichView210;
  ARVData: TCustomRVData957;
  AItem: TCustomRVItemInfo; AState: TRVItemDrawStates;
  const ARect: TRVCoordRect998; ACanvas: TCanvas) of object;
```

FireMonkey:

type

```
TRVAfterDrawImageEvent = procedure (Sender: TCustomRichView210;
```

```
ARVData: TCustomRVData957;
AItem: TCustomRVItemInfo; AState: TRVItemDrawStates;
const ARect: TRVCoordRect998; ACanvas: TRVFMXCanvas961) 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.OnAfterPrintImage⁸²⁹

6.1.1.4.3 TRichView.OnAssignImageFileName

Occurs when inserting an image file in TRichView.

type

```
TRVAssignImageFileNameEvent = procedure (Sender: TCustomRichView210;
var FileName: TRVUnicodeString1032) of object;
```

property OnAssignImageFileName: TRVAssignImageFileNameEvent;

(introduced in version 16; changed in version 18)

This event occurs if *rvoAssignImageFileNames* is included in Options²⁴⁰.

In this event, you can modify **FileName** (for example, change a full path to a relative path).

This event may happen on:

- RTF loading³²⁶ (reading external images),
- DocX loading³¹⁸ (reading external images),
- Markdown loading³²⁵ (reading external images),
- HTML loading (using importer components),
- pasting files⁵³⁴, accepting files and links to images as a result of drag&drop¹¹⁸,
- RichViewActions (actions for inserting images and assigning background images of tables).

The following properties are set:

- table.BackgroundImageFileName⁸⁵⁰
- cell.BackgroundImageFileName⁸⁹⁷
- rvesplmageFileName¹⁰⁰⁵ extra item properties of pictures¹⁶³ and hot-pictures¹⁶⁶.

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⁽²²⁷⁾ = *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⁽²²⁷⁾.

See also methods extracting information from CheckpointData:

- GetCheckpointInfo⁽²⁹¹⁾;
- GetCheckpointXY⁽²⁹²⁾;
- GetCheckpointYEx⁽²⁹³⁾;
- GetCheckpointItemNo⁽²⁹¹⁾;
- GetCheckpointNo⁽²⁹²⁾.

See also

- Checkpoints⁽⁸⁷⁾.

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⁽³⁸¹⁾

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**⁽²⁵³⁾.RVData (must be typecasted to TCustomRVData⁽⁹⁵⁷⁾) and **ItemNo** parameter. **ItemNo** is an index of item (containing **ctrl**) in the document **Sender.Style.RVData** (it may be RVData⁽²⁴⁷⁾ of the main RichView (**Sender**), cell⁽⁸⁹⁵⁾, or RVData⁽²⁴⁷⁾ of cell inplace editor).

ControlAction	Meaning
<i>rvcaAfterRVFLoad</i>	The control was created and loaded from RVF (file, stream, database field or Clipboard);
<i>rvcaDestroy</i>	The control will be destroyed soon (removing items from RichView, or destroying RichView itself)
<i>rvcaMoveToUndoList</i>	The control was moved to undo/redo buffer. It will not be displayed, but it will not be destroyed yet.
<i>rvcaMoveFromUndoList</i>	The control was moved from undo/redo list back to the editor.
<i>rvcaDestroyInUndoList</i>	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 ;

ControlAction	Meaning
<i>rvcaBeforeRVFSave</i>	Before saving to RVF
<i>rvcaAfterRVFSave</i>	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⁽³⁸⁰⁾.

6.1.1.4.7 TRichView.OnCopy

Occurs when copying to the Clipboard

property OnCopy: TNotifyEvent;

(Introduced in version 1.4)

Occurs inside CopyDef⁽²⁸¹⁾ and Copy⁽²⁸¹⁾ 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⁽⁵⁸⁴⁾;
- Working with Clipboard⁽¹⁰⁸⁾.

6.1.1.4.8 TRichView.OnGetItemCursor

Allows defining custom custom cursors for items.

type

```
TRVGetItemCursorEvent = procedure (Sender: TCustomRichView(210);
  RVData: TCustomRVData(957); 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⁽²⁴⁷⁾, or table cell, or cell inplace editor's RVData⁽²⁴⁷⁾)

ItemNo – index of the item inside **RVData**.

Output Parameter

Cursor – cursor to display.

See also:

- OnItemHint⁽³⁸¹⁾.

6.1.1.4.9 TRichView.OnGetSpellingSuggestions

This event requests a list of suggestions for the misspelled word.

type

```
TRVGetSpellingSuggestionsEvent = procedure(  
    Sender: TCustomRichView(210);  
    const AWord: TRVUnicodeString(1032); 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⁽⁴⁶¹⁾.

If a platform spelling check service is used (see CheckSpelling⁽⁴⁷²⁾ 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⁽⁴⁸⁸⁾ to add items in any popup menu.

Input parameters:

AWord is a word to correct.

StyleNo is a style of text item⁽¹⁶¹⁾ containing this word (index in the collection **Sender.Style**⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾).

Output parameter:

ASuggestions, a list of suggestions to replace **AWord** in the document.

See also:

- Live spelling check in TRichView⁽¹³²⁾;
- OnSpellingCheck⁽⁴⁰⁹⁾ 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(1032);  
        BackgroundColor: TRVColor(996); 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⁽¹⁶³⁾, *hot-pictures*⁽¹⁶⁶⁾,
- *bullets*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾,
- list markers⁽¹⁷⁴⁾ (paragraph bullets) with images,
- tables⁽¹⁷³⁾ (saving background image⁽⁸⁴⁹⁾),
- for background image⁽²²²⁾ and cells' background images⁽⁸⁹⁶⁾.

There is more easy-to-use event: OnSaveImage2⁽⁴⁰²⁾.

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 using RVGraphicHandler⁽⁹⁷²⁾. File names of saved images are composed as ImagesPrefix + Number + '.' + FileExtension, where

- ImagesPrefix – HTMLSaveProperties⁽²³⁷⁾.ImagesPrefix⁽⁴³⁶⁾;
- Number – some number making this file name unique (image files can override existing ones or not, depending on *rvhtmlsioOverrideImages* in HTMLSaveProperties⁽²³⁷⁾.ImageOptions⁽⁴³⁴⁾).

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⁽²²⁶⁾ /TRVStyle.Color⁽⁶³⁵⁾), color of cell (TRVTableCellData.Color⁽⁹⁰⁰⁾) or color of paragraph background (TRVBackgroundRect.Color⁽⁷⁴²⁾).

You can define formats that must not be converted to Jpeg⁽¹⁰⁷⁰⁾.

Input Parameters

RVData – document containing item that is being saved to image; it can be **Sender.RVData**⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, 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⁽³³⁸⁾).

BackgroundColor – color of background below the item (can be ignored, if you can preserve transparency of image)

Location – if *rvhtmlsioUseItemImageFileNames* is included in HTMLSaveProperties⁽²³⁷⁾.ImageOptions⁽⁴³⁴⁾, this is a path to the image defined in the *rvspltImageFileName* item property⁽¹⁰⁰⁵⁾, 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*⁽²³⁷⁾.*ImageOptions*⁽⁴³⁴⁾ and value of *rvshtmlImageFileName* item property⁽¹⁰⁰⁵⁾ is not empty, this value is written in HTML (relative to **Path**); otherwise
- *OnSaveImage2*⁽⁴⁰²⁾ 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;
- **RVDData**.*GetNextFileName* – function returning image file name with the given image prefix and extension:

```
function GetNextFileName(  
    const ImagesPrefix, Path, Ext: TRVUnicodeString(1032);  
    var imgSaveNo: Integer; OverrideFiles: Boolean): TRVUnicodeString(1032);
```

where

ImagesPrefix – the first part of image file name (for example, you can set it to *HTMLSaveProperties*⁽²³⁷⁾.*ImagesPrefix*⁽⁴³⁶⁾);

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*⁽¹⁰⁵⁷⁾.*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*⁽³⁴²⁾.

See also events:

- *OnSaveImage2*⁽⁴⁰²⁾;
- *OnSaveItemToFile*⁽⁴⁰⁴⁾.

See also

- *Export to HTML*⁽¹²⁷⁾.

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: TRVUnicodeString1032;
  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 OnAfterOleDrop⁵⁷⁰ 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: TRVUnicodeString1032; Width, Height: Integer;
  var Graphic: TRVGraphic970) 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* ⁽⁴⁷⁵⁾.

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* ⁽⁵⁷⁰⁾ events.

Example: downloading images from http location using Indy (IdHTTP1: TIdHTTP)

```
uses RVFuncs;  
  
{ TMyRichView.OnImportPicture }  
procedure TMyForm.MyRichViewImportPicture(Sender: TCustomRichView;  
  const Location: TRVUnicodeString(1032); Width, Height: Integer;  
  var Graphic: TRVGraphic(970));  
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(1057).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: TCustomRichView210;  
ItemAction: TRVItemAction; Item: TCustomRVItemInfo844;  
var Text: TRVUnicodeString1032; RVDData: TCustomRVDData957) of object;
```

property OnItemAction: TRVItemActionEvent;

(introduced in version 1.7; modified in v18)

Parameters

Item – item⁸⁴⁴ for which this event is generated.

Text – text associated with this item. If this is a text item, this text is displayed in TRichView.

RVDData – document containing Item (it can be **Sender.RVDData**²⁴⁷, table cell⁸⁹⁵, or RVDData of cell inplace-editor).

ItemAction identifies operation that calls this event.

ItemAction	Meaning
<i>rviaInserting</i>	The event occurs before the item is inserted in Sender (in RVDData). You can modify Text . This event also occurs when moving items to cell inplace editor and back to the cell.
<i>rviaInserted</i>	The event occurs after the item is inserted in the Sender (in RVDData). Ignore Text parameter. This event also occurs when moving items to cell inplace editor and back to the cell.
<i>rviaMovingToUndoList</i>	The event occurs when item is moved (from RVDData) to undo list. Ignore Text parameter.
<i>rviaDestroying</i>	Occurs when item is destroyed. Ignore Text parameter. If the item is in undo/redo list, RVDData is <i>nil</i> . Otherwise, this is an object containing this item.
<i>rviaTextModifying</i>	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: TRVUnicodeString1032; 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:

- OnControlAction³⁷³.

See also methods:

- GetItem²⁹⁶;
- GetItemNo³⁰¹.

6.1.1.4.14 TRichView.OnItemHint

Allows to define custom popup hints for items.

```

type
  TRVItemHintEvent = procedure (Sender: TCustomRichView210;
    RVData: TCustomRVData957; ItemNo: Integer;
    var HintText: TRVUnicodeString1032) of object;

```

```

property OnItemHint: TRVItemHintEvent;

```

This event occurs when mouse pointer moves over the item.

Parameters

RVData – document containing this item (RichView.RVData²⁴⁷, or table cell, or cell inplace editor's RVData²⁴⁷)

ItemNo – index of the item inside **RVData**.

HintText – on input, it's a value of *rvespHint* item property¹⁰⁰⁵. On output, it's a message that will be displayed.

Item popup hint can be displayed if *rvoShowItemHints* is in RichView.Options²⁴⁰ and RichView.ShowHint=True.

Cell Hint⁹⁰¹s cannot be altered in this event.

See also:

- OnGetItemCursor³⁷⁴.

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(210);
  RVData: TCustomRVData(957); ItemNo: Integer;
  var HintText: TRVUnicodeString(1032));
var Tag: TRVTag(1029);
  VAlign: TRVVAlign(1033);
  Name: TRVUnicodeString(1032);
  gr: TGraphic;
begin
  if RVData.GetItem(296)(ItemNo).GetBoolValueEx(
    rvbpJump, Sender.Style(253)) then
  begin
    if HintText<>' ' then
      HintText := HintText+#13;
      HintText := HintText + 'Target: '+RVData.GetItemTag(302)(ItemNo);
    end;
    if (RVData.GetItemStyle(302)(ItemNo)=rvsPicture(1059)) or
      (RVData.GetItemStyle(302)(ItemNo)=rvsHotPicture(1059)) then
    begin
      if HintText<>' ' then
        HintText := HintText+#13;
        RVData.GetPictureInfo(310)(ItemNo, Name, gr, VAlign, Tag);
        HintText := HintText+gr.ClassName+
          ' ('+IntToStr(gr.Width)+' , '+IntToStr(gr.Height)+' )';
      end;
    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⁽²³⁶⁾, the next link has **id**=FirstJumpNo⁽²³⁶⁾+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⁽⁴⁷⁵⁾) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly⁽⁴⁸⁰⁾=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 event can also be initiated using keyboard, if `TabNavigation` ⁽²⁵⁵⁾ `<>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 (957);  
    s: String;  
begin  
    MyRichView.GetJumpPointLocation (304)(id, RVData, ItemNo);  
    s := RVData.GetItemTag (302)(ItemNo);  
    Application.MessageBox(PChar(s), 'Link', MB_OK or MB_ICONINFORMATION);  
end;
```

See also events:

- `OnRVMouseMove` ⁽³⁹⁵⁾.

See also:

- `RichView` `hypertext` ⁽⁹²⁾.

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 (210);  
    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` ⁽²³²⁾ property)
- by `TDBRichView` ⁽⁵⁹⁴⁾ and `TDBRichViewEdit` ⁽⁶⁰⁶⁾ 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` ⁽⁴⁶¹⁾.`OnSaveCustomFormat` ⁽⁵⁸⁵⁾

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⁽²³²⁾ property)
- by TDBRichView⁽⁵⁹⁴⁾ and TDBRichViewEdit⁽⁶⁰⁶⁾ components

This event allows to preprocess the loaded document before displaying.

See also:

- OnNewDocument⁽³⁸⁴⁾.

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⁽²³²⁾ property)
- by TDBRichView⁽⁵⁹⁴⁾ and TDBRichViewEdit⁽⁶⁰⁶⁾ 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⁽³⁸⁴⁾.

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`⁽⁷⁸²⁾, `OnPagePostpaint`⁽⁷⁸¹⁾).

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`⁽⁸¹⁶⁾, `VScrollPos`⁽⁸¹⁹⁾) 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⁽¹²⁴⁾;
- reading/writing HTML;
- writing DocX;
- reading/writing a plain ANSI or Unicode text;
- when calling `TRVOfficeConverter.ImportRV`⁽⁸⁰²⁾ and `ExportRV`⁽⁸⁰⁰⁾.

Note: the event is called only if the document/source file is large enough.

See also:

- TRVOfficeConverter.OnConverting⁸⁰³

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: TRVUnicodeString1032;
  Tag: TRVTag;
  StyleNo: Integer;
end;

TRVFieldContext = record
  FirstTime: Boolean;
  ParaNo: Integer;
  FieldResultText: TRVUnicodeString1032;
end;

```

type

```

TRVReadFieldEvent = procedure (
  Sender: TCustomRichView210;
  const FieldCode: TRVUnicodeString1032;
  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** $\langle \rangle$ *nil*, it must be an object of class inherited from TCustomRVItemInfo⁸⁴⁴. 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, TRVLabelItemInfo⁹¹² has **TextStyleNo**⁹¹⁵ property), you can assign *rvsDefStyle*¹⁰⁵⁹ 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** \geq 0, it defines a text style of this text (index in *Style*²⁵³.*TextStyles*⁶⁵⁴ collection). If **FieldResult.StyleNo** $<$ 0 (or is equal to *rvsDefStyle*¹⁰⁵⁹), the current RTF/DocX text attributes are used. **FieldResult.Text** may be multiline, but cannot contain tab (#9) characters. If **FieldResult.Tag** $\langle \rangle$ "", this tag⁹¹ 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*²⁵³.*ParaStyles*⁶⁴⁸ 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,²⁵⁶ you can assign a text style with the proper value of *ParaStyleTemplateId*⁷⁰⁵.

6.1.1.4.22 TRichView.OnReadHyperlink

Occurs when reading hyperlinks (for example, from RTF file)

type

```
TRVReadHyperlink =
  procedure (Sender: TCustomRichView;
    const Target, Extras: TRVUnicodeString1032;
    DocFormat: TRVLoadFormat1015;
    var StyleNo: Integer; var ItemTag: TRVTag1029;
    var ItemName: TRVUnicodeString1032) of object;
```

property OnReadHyperlink: TRVReadHyperlink;

(changed in version 18)

By default, hyperlink targets is saved in the item tag⁹¹. 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**='\"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 property¹⁰⁰⁵, and adds '#' to the beginning of links to bookmarks.

DocFormat – format of loaded document.

DocFormat	Meaning
<i>rvlfRTF</i>	The event occurs when reading RTF files (or streams) containing hyperlinks (RTF loading, pasting, or accepting on drag&drop).
<i>rvlfDocX</i>	The event occurs when reading DocX (Microsoft Word Document) files containing hyperlinks
<i>rvlfURL</i>	The event occurs: 1) on drag&drop, when accepting URL dropped to TRichViewEdit ⁴⁶¹ (<i>rvddURL</i> must be in AcceptDragDropFormats ⁴⁷⁰). 2) when pasting from the Clipboard, when pasting URL to TRichViewEdit ⁴⁶¹ (<i>rvddURL</i> must be in AccepPasteFormats ⁴⁷⁰).
<i>rvlfHTML</i>	The event occurs when reading HTML files containing hyperlinks.

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⁽⁶⁵⁴⁾ of this text. You can modify it to point to another (hypertext⁽⁷¹⁴⁾) style. If **DocFormat=rvIfURL**, 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⁽⁹¹⁾ 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⁽⁹¹⁾.

```

procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString(1032); DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag(1029);
  var ItemName: TRVUnicodeString(1032));
begin
  ItemTag := Target;
end;

```

More complicated example

This example assumes that you included *rvddURL* in **AcceptDragDropFormat**⁽⁴⁷⁰⁾ and want to accept hyperlinks. Besides, it's assumed that "Allow adding styles dynamically" is set in the component editor⁽²¹⁸⁾ (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(1032); DocFormat: TRVLoadFormat(1015);
  var StyleNo: Integer; var ItemTag: TRVTag(1029);
  var ItemName: TRVUnicodeString(1032));
var FontStyle: TFontInfo(712);
begin
  ItemTag := Target;
  if DocFormat=rvddURL then
  begin
    FontStyle := TFontInfo(712).Create(nil);
    FontStyle.Assign(Sender.Style(253).TextStyles(654)[StyleNo]);
    FontStyle.Color(701) := clBlue;
    FontStyle.Style(708) := FontStyle.Style + [fsUnderline];
    FontStyle.Jump(714) := True;
  end;

```

```

StyleNo := Sender.Style.FindTextStyle661(FontStyle);
FontStyle.Free;
end;
end;

```

The same example using RichViewActions:

```

procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
const Target, Extras: TRVUnicodeString1032; DocFormat: TRVLoadFormat1015;
var StyleNo: Integer; var ItemTag: TRVTag1029;
var ItemName: TRVUnicodeString1032);
begin
ItemTag := Target;
if DocFormat=rvddURL then
StyleNo := rvActionInsertHyperlink1.GetHyperlinkStyleNo(
Sender as TRichViewEdit);
end;

```

See also events:

- OnWriteHyperlink⁴¹¹.

See also properties of RVRTFReadProperties²⁴⁶:

- BasePathLinks⁴⁵¹.

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: TRVUnicodeString1032;
DocFormat: TRVLoadFormat;
var StyleNo: Integer; var ItemTag: TRVTag1029;
var ItemName: TRVUnicodeString1032) 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 OnReadField³⁸⁶, 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 *rvIfRTF*.

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*⁽⁸⁴⁴⁾)
- if the item is a text, it is an index of style⁽⁶⁵⁴⁾ 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⁽⁹¹⁾ of this item. You can store **FieldName** here. However, please note that targets of hyperlinks⁽⁹²⁾ 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(1032); DocFormat: TRVLoadFormat(1015);
  var StyleNo: Integer; var ItemTag: TRVTag(1029);
  var ItemName: TRVUnicodeString(1032));
begin
  if StyleNo >= 0 then
    ItemName := '{' + FieldName + '}';
end;

```

6.1.1.4.24 TRichView.OnRVDbClick

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⁽²⁴⁰⁾)

```

type
  TRVDbClickEvent =
    procedure(Sender: TCustomRichView(210);

```

```
ClickedWord: TRVUnicodeString(1032);
Style: Integer) of object;
```

```
property OnRVDbClick: TRVDbClickEvent;
```

(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⁽⁸⁵⁾). In this case **ClickedWord** contains a name of this item.

If **Style**>=0, then this is a text item⁽¹⁶¹⁾, and **Style** is an index in the collection of text styles (Style⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾). 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⁽²²⁸⁾ property.

This event is not generated when the user clicks not on an item (i.e. on a background)

See also events:

- OnRVRightClick⁽³⁹⁷⁾.

See also properties:

- Delimiters⁽²²⁸⁾.

6.1.1.4.25 TRichView.OnRVFControlNeeded

Occurs when reading inserted control⁽¹⁶⁸⁾ from RVF file or stream, and this control was saved without its "body".

type

```
TRVFControlNeededEvent =
procedure(Sender: TCustomRichView(210); Name: TRVUnicodeString(1032);
const Tag: TRVTag(1029); 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⁽²⁴⁷⁾, none of control properties are saved. Only the item name (value returned by GetItemText⁽³⁰³⁾) and *tag* (value returned by GetItemTag⁽³⁰²⁾) 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⁽³⁰³⁾; do not confuse with Name property of controls, see Item types⁽⁸⁵⁾).

Tag – *tag* of the control's item (this is a value returned by `GetItemTag`⁽³⁰²⁾; do not confuse with `Tag` property of controls, see "Tags"⁽⁹¹⁾).

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⁽¹²⁴⁾, contains example.

6.1.1.4.26 TRichView.OnRVFImageListNeeded

Occurs when reading *bullet*⁽¹⁷⁰⁾, *hotspot*⁽¹⁷²⁾ or list level with imagelist picture⁽⁷⁵⁴⁾ from RVF or XML (by TRichViewXML).

type

```
TRVFImageListNeededEvent =
  procedure (Sender: TCustomRichView(210);
    ImageListTag: Integer;
    var Ail: TCustomImageList) of object;
```

property OnRVFImageListNeeded: TRVFImageListNeededEvent;

If you do not process this event, TRichView is not able to read *bullets*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾ 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⁽⁹¹⁾, 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⁽¹²⁴⁾, contains example.

6.1.1.4.27 TRichView.OnRVFPictureNeeded

Occurs when reading picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾ from RVF file or stream, and this picture was saved without its "body"

type

```
TRVFPictureNeededEvent = procedure (Sender: TCustomRichView(210);
  const ItemName: TRVUnicodeString(1032); Item: TRVNonTextItemInfo(844);
  Index1, Index2: Integer; var gr: TRVGraphic(970)) 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*⁽²⁴⁷⁾, images are not saved. Only item name (value returned by *GetItemText*⁽³⁰³⁾) and *tag*⁽⁹¹⁾ (value returned by *GetItemTag*⁽³⁰²⁾) 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*⁽¹⁶³⁾ and *hot-pictures*⁽¹⁶⁶⁾;
- *table*⁽¹⁷³⁾ background images⁽⁸⁴⁹⁾ (even if the saved table did not have a background image initially)
- *table cell background images*⁽⁸⁹⁶⁾ (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*⁽³⁰³⁾). 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*⁽⁸⁴⁴⁾.

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⁽¹²⁴⁾, 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(210); Button: TMouseButton;
    Shift: TShiftState; ItemNo, X, Y: TRVCoord(998)) 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: TCustomRichView210;
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: TRVCoord998);
var LRVDData: TCustomRVFormattedData957;
  LItemNo, LOffs: Integer;
  pt: TRVCoordPoint998;
begin
  pt := MyRichView.ClientToDocument280(Point(X,Y));
  if not MyRichView.GetItemAt297(pt.X, pt.Y, LRVDData, LItemNo, LOffs,
    True) then
    exit;
  // item is the LItemNo-th item in LRVDData
  // example of using: LRVDData.GetItemStyle302(LItemNo)
  ..
end;

```

Note: if you want to create code responding on clicking on item, it's recommended to use OnRVMouseUp³⁹⁶ instead of OnRVMouseDown.

See also events:

- OnRVMouseUp³⁹⁶.

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**=FirstJumpNo²³⁶, the next link has **id**=FirstJumpNo²³⁶+1, and so on).

In editor, this numeration is available only when the editor works in hypertext mode:

- while user holds **Ctrl** key (see EditorOptions⁴⁷⁵) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly⁴⁸⁰=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`⁽³⁸²⁾.

See also:

- `RichView` hypertext overview⁽⁹²⁾, 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(210);
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: TRVCoord(998)) 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 `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(210);
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: TRVCoord(998));
var LRVDData: TCustomRVFormattedData(957);
  LItemNo, LOffs: Integer;
  pt: TRVCoordPoint(998);
begin
  pt := MyRichView.ClientToDocument(280)(Point(X,Y));
  if not MyRichView.GetItemAt(297)(pt.X, pt.Y, LRVDData, LItemNo, LOffs,
    True) then
    exit;
  // item is the LItemNo-th item in LRVDData
  // example of using: LRVDData.GetItemStyle(302)(LItemNo)
  ..
end;
```

See also events:

- `OnRVMouseDown`⁽³⁹⁴⁾;

- OnRVRightClick⁽³⁹⁷⁾.

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(210);
  ClickedWord: TRVUnicodeString(1032); Style, X, Y: TRVCoord(998)) of object;
```

property OnRVRightClick: TRVRightClickEvent;

This event is almost obsolete. There are more convenient events:

- OnRVMouseDown⁽³⁹⁴⁾,
- OnRVMouseUp⁽³⁹⁶⁾.

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*⁽¹⁰⁵⁹⁾). In this case **ClickedWord** contains name of this item (see Item types⁽⁸⁵⁾).

If **Style**>=0, this is a text item, and **Style** is an index in the collection of text styles (Style⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾). 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⁽²²⁸⁾ 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⁽³⁹⁴⁾, OnRVMouseUp⁽³⁹⁶⁾):

- 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:

- OnRVDbClick⁽³⁹¹⁾.

See also properties:

- Delimiters⁽²²⁸⁾.

6.1.1.4.32 TRichView.OnSaveComponentToFile

Occurs when TRichView wants to save inserted control⁽¹⁶⁸⁾ in text, Markdown, RTF, HTML, or DocX file (or stream).

type

```
TRVSaveComponentToFileEvent = procedure(
  Sender: TCustomRichView(210); Path: TRVUnicodeString(1032);
  SaveMe: TPersistent; SaveFormat: TRVSaveFormat(1020);
  var OutStr: TRVUnicodeString(1032)) 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`⁽³⁴²⁾ method).

Additional input parameters

Sender.Style⁽²⁵³⁾.RVDData, **Sender.Style**⁽²⁵³⁾.ItemNo identify the item to save. RVDData – document containing item that is being saved; it can be **Sender.RVDData**⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, or RVDData of cell inplace-editor. ItemNo – index of this item inside RVDData.

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(1032);
  SaveMe: TPersistent; SaveFormat: TRVSaveFormat;
  var OutStr: TRVUnicodeString(1032));
var bmp: TBitmap;
  ImageFileName: TRVUnicodeString(1032);
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;
  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 ⁽⁴⁰⁴⁾;
- Saving and loading ⁽¹²²⁾;
- Export to HTML ⁽¹²⁷⁾.

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(1032))
of object;

```

property OnSaveDocXExtra: TRVSaveDocXExtraEvent;

(changed in version 18)

This event occurs when saving DocX (SaveDocX⁽³³³⁾, SaveDocXToStream⁽³³⁴⁾).

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
<i>rv_docxs_TextStyle</i>	Occurs when saving a text style. Index1 is an index of the text style in TextStyles ⁽⁶⁵⁴⁾ collection, Obj is a text style object ⁽⁷¹²⁾	<w:rPr>	document files*

Area	Meaning	Parent XML Element	XML file
<i>rv_docxs_ParaStyle</i>	Occurs when saving a paragraph style. Index1 is an index of the paragraph style in ParaStyles ⁽⁶⁴⁸⁾ collection, Obj is RVData ⁽⁹⁵⁷⁾ containing this paragraph. Index2 is an index of the first item of this paragraph.	<w:pPr>	document files*
<i>rv_rtfs_ListStyle</i>	Occurs when saving a list style and list levels. Index1 is an index of the list style in ListStyles ⁽⁶⁴⁶⁾ . 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 ⁽⁷³¹⁾	if Index2 = -1: <w:abstractNum> ; if Index2 >= 0: <w:lvl>	word/numbering.xml
<i>rv_docxs_StyleTemplate</i>	Occurs when saving a style template as a style sheet item. Index1 is an index of the style template in StyleTemplates ⁽⁶⁵²⁾ . Obj is a style template ⁽⁷³³⁾ .	<w:style>	word/styles.xml
<i>rv_docxs_StyleTemplate Text</i>	Occurs when saving text attributes of a style template inside a style sheet item. Index1 is an index of the style template in StyleTemplates ⁽⁶⁵²⁾ . Obj is a style template ⁽⁷³³⁾ .	<w:rPr> inside <w:style>	word/styles.xml
<i>rv_docxs_StyleTemplate Para</i>	Occurs when saving paragraph attributes of a style template inside a style sheet item. Index1 is an index of the style template in StyleTemplates ⁽⁶⁵²⁾ . Obj is a style template ⁽⁷³³⁾ .	<w:pPr> inside <w:style>	word/styles.xml
<i>rv_docxs_CellProps</i>	Occurs when saving table cell properties Index1 = row index, Index2 = column index, Obj is the table ⁽⁸⁴⁶⁾ .	<w:tcPr>	document files*
<i>rv_docxs_RowProps</i>	Occurs when saving table row properties.	<w:trPr>	document files*

Area	Meaning	Parent XML Element	XML file
	Index1 = row index, Obj is the table ⁽⁸⁴⁶⁾ .		
<i>rv_docxs_TableProps</i>	Occurs when saving table properties. Obj is the table ⁽⁸⁴⁶⁾ .	<w:tblPr>	document files*
<i>rv_docxs_SectProps</i>	Occurs when saving page properties.	<w:sectPr>	word/document.xml
<i>rv_docxs_Settings</i>	Occurs when saving document settings.	<w:settings>	word/settings.xml
<i>rv_docxs_CoreProperties</i>	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 ⁽²³⁰⁾.

See also events:

- OnSaveRTFExtra ⁽⁴⁰⁶⁾.
- OnSaveHTMLExtra ⁽⁴⁰¹⁾.

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(1032)) of object;
```

property OnSaveHTMLExtra: TRVSaveHTMLExtraEvent;

(changed in version 18)

This event occurs when saving HTML.

If HTMLSaveProperties ⁽²³⁷⁾.HTMLSavingType ⁽⁴³³⁾ = *rvhtmlstNormal*, **CSSVersion** parameter = True.

If HTMLSaveProperties ⁽²³⁷⁾.HTMLSavingType ⁽⁴³³⁾ = *rvhtmlstSimplified*, **CSSVersion** parameter = False.

Text assigned to **HTMLCode** parameter will be saved in HTML.

Area can be:

Area	HTMLCode is inserted...
<i>rv_thms_Head</i>	between <head> and </head>
<i>rv_thms_BodyAttribute</i>	between <body and >
<i>rv_thms_Body</i>	just after <body>
<i>rv_thms_End</i>	just before </body>

Example

```

procedure TMyForm.MyRichViewSaveHTMLExtra(
  Sender: TCustomRichView; Area: TRVHTMLSaveArea;
  CSSVersion: Boolean; var HTMLCode: TRVUnicodeString1032);
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:

- OnHTMLSaveImage³⁷⁵;
- OnSaveImage2⁴⁰²;
- OnSaveParaToHTML⁴⁰⁶;
- OnSaveRTFExtra⁴⁰⁶;
- OnSaveDocXExtra³⁹⁹.

See also:

- Export to HTML¹²⁷.

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: TRVGraphic970; SaveFormat: TRVSaveFormat1020;
  const Path, ImagePrefix: TRVUnicodeString1032;
  var ImageSaveNo: Integer; var Location: TRVUnicodeString1032;
  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`⁽³⁷⁵⁾. 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`⁽²³⁷⁾.`ImageOptions`⁽⁴³⁴⁾ and value of `rvshtmlImageFileName` item property⁽¹⁰⁰⁵⁾ is not empty;
- `DoDefault` parameter of `OnHTMLSaveImage`⁽³⁷⁵⁾ was set to *False* for this image.

For Markdown, **OnSaveImage2** is not called if:

- `rvmdsoUseItemImageFileNames` is included in the `MarkdownProperties`⁽²³⁹⁾.`SaveOptions`⁽⁴⁴⁸⁾ and value of `rvshtmlImageFileName` item property⁽¹⁰⁰⁵⁾ 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`⁽³³⁸⁾/`SaveMarkdownToStream`⁽³⁴¹⁾)

ImagePrefix – recommended beginning of image file name (`HTMLSaveProperties`⁽²³⁷⁾.`ImagesPrefix`⁽⁴³⁶⁾ or `MarkdownProperties`⁽²³⁹⁾.`ImagesPrefix`⁽⁴⁴⁵⁾)

ImageSaveNo – value that you can use to create unique file names (see `GetNextFileName` method in the description of `OnHTMLSaveImage`⁽³⁷⁵⁾).

"Hidden" input parameters:

Sender.Style.RVData (must be typecasted to `TCustomRVData`⁽⁹⁵⁷⁾) – document containing the item that is being saved to an image; it can be **Sender.RVData**⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, 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*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾, list markers⁽¹⁷⁴⁾ with images), `TRichView` creates a temporal `TBitmap` and passes it to this event.

See also events:

- OnHTMLSaveImage⁽³⁷⁵⁾;
- OnSaveItemToFile⁽⁴⁰⁴⁾;
- OnImportPicture⁽³⁷⁸⁾.

See also:

- Export to HTML⁽¹²⁷⁾.

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(1032);
  RVDData: TCustomRVDData(957); ItemNo: Integer;
  SaveFormat: TRVSaveFormat(1020); var OutStr: TRVUnicodeString(1032);
  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⁽³⁷⁵⁾,
- OnSaveImage2⁽⁴⁰²⁾,
- OnSaveComponentToFile⁽³⁹⁷⁾.

These events are easier to use than this event.

Input parameters:

Path – path to the output file;

RVDData, **ItemNo** – item to save. **RVDData** – document containing item that is being saved; it can be **Sender.RVDData**⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, or RVDData of cell inplace-editor. **ItemNo** – index of this item inside **RVDData**.

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
<i>True</i>	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.
<i>False</i>	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 ⁽¹⁷⁷⁾ 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: TRVUnicodeString1032;
  RVData: TCustomRVData957; ItemNo: Integer;
  SaveFormat: TRVSaveFormat1020;
  var OutStr: TRVUnicodeString1032; var DoDefault: Boolean);
begin
  if (RVData.GetItemStyle302(ItemNo)=rvsPicture1059) 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(957); ItemNo: Integer;
  ParaStart, CSSVersion: Boolean;
  var HTMLCode: TRVUnicodeString(1032)) 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 <p>, <div> or),
- *False*, if it will be inserted in the end (after </p>, </div> 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**⁽²³⁷⁾ for the last paragraph).

RVData can be **Sender.RVData**⁽²⁴⁷⁾, or table cell, or cell inplace editor's **RVData**⁽²⁴⁷⁾.

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(1032));
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⁽⁴⁰¹⁾;
- Export to HTML⁽¹²⁷⁾.

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(1032)) of object;

```

property OnSaveRTFExtra: TRVSaveRTFExtraEvent;

(changed in version 18)

This event occurs when saving RTF (SaveRTF⁽³⁴³⁾, SaveRTFToStream⁽³⁴⁴⁾).

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
<i>rv_rtfs_TextStyle</i>	Occurs when saving text style. Index1 is an index of the text style in TextStyles ⁽⁶⁵⁴⁾ collection, Obj is a text style object ⁽⁷¹²⁾ If InStyleSheet=True , this style is saved as a part of style sheet (when RTF is saved with <i>rvrtfSaveStyleSheet</i> ⁽²⁴⁵⁾ option)
<i>rv_rtfs_ParaStyle</i>	Occurs when saving paragraph style. Index1 is an index of the paragraph style in ParaStyles ⁽⁶⁴⁸⁾ collection, Obj is a paragraph style object ⁽⁷¹⁸⁾ If InStyleSheet=True , this style is saved as a part of style sheet (when RTF is saved with <i>rvrtfSaveStyleSheet</i> ⁽²⁴⁵⁾ option)
<i>rv_rtfs_ListStyle</i>	Occurs when saving list style and list levels. Index1 is an index of the list style in ListStyles ⁽⁶⁴⁶⁾ . 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 ⁽⁷³¹⁾
<i>rv_rtfs_StyleTemplate</i>	Occurs when saving a style template as a style sheet item. Index1 is an index of the style template in StyleTemplates ⁽⁶⁵²⁾ . Index2 = -1 Obj is a style template ⁽⁷³³⁾ .
<i>rv_rtfs_StyleTemplateText</i>	Occurs when saving a style template as an additional style sheet item (containing only character properties), linked the the main style

Area	Meaning
	sheet item. Such additional style sheet items are created for style templates having Kind ⁽⁷³⁵⁾ = <i>rvstkParaText</i> . Index1 is an index of the style template in StyleTemplates ⁽⁶⁵²⁾ . Index2 = -1 Obj is a style template ⁽⁷³³⁾ .
<i>rv_rtfs_CellProps</i>	Occurs when saving table cell properties. Index1 = row index, Index2 =column index, Obj is the table ⁽⁸⁴⁶⁾ .
<i>rv_rtfs_RowProps</i>	Occurs when saving table row properties. Index1 = row index, Obj is the table ⁽⁸⁴⁶⁾ .
<i>rv_rtfs_Doc</i>	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(993));
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⁽²³⁰⁾.

See also events:

- OnSaveDocXExtra⁽³⁹⁹⁾;
- OnSaveHTMLExtra⁽⁴⁰¹⁾.

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⁽³⁴⁸⁾;
- Deselect⁽²⁸⁶⁾;

- `SetSelectionBounds`⁽³⁶⁵⁾;
- (and sometimes `SelectionExists`⁽³⁴⁹⁾).

See also:

- Selection in `RichView`⁽¹⁰⁷⁾.

6.1.1.4.40 `TRichView.OnSpellingCheck`

Allows to mark misspelled words.

type

```
TRVSpellingCheckEvent = procedure (Sender: TCustomRichView;
  const AWord: TRVUnicodeString(1032); 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⁽¹⁶¹⁾ containing this word (index in the collection `Sender.Style`⁽²⁵³⁾.`TextStyles`⁽⁶⁵⁴⁾).

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(1032); StyleNo: Integer;
  var Misspelled: Boolean);
begin
  Misspelled := not MyRVAddictSpell3.WordAcceptable(AWord);
end;
```

This event is called in thread context.

Using with `TRVSpellChecker`⁽⁷⁸⁴⁾

You have two options:

- register this `TRichView` component using `RVSpellChecker.RegisterEditor`⁽⁷⁹²⁾ method (and this event will be processed automatically), or
- process this event and use `RVSpellChecker.IsWordValid`⁽⁷⁹¹⁾ method.

FireMonkey note:

If the current platform provides a spelling checking service, TRichViewEdit can use it (see CheckSpelling⁽⁴⁷²⁾ 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⁽¹³²⁾.
- OnGetSpellingSuggestions⁽³⁷⁵⁾ [FMX]

6.1.1.4.41 TRichView.OnStyleTemplatesChange

Occurs when the collection Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾ was changed.

property OnStyleTemplatesChange: TNotifyEvent;

(introduced in version 14)

This event may be called only if UseStyleTemplates⁽²⁵⁶⁾ = True.

It is called:

- when loading or inserting RVF, RTF, DocX files or streams (and StyleTemplateInsertMode⁽²⁵⁴⁾ <>rvstimIgnoreSourceStyleTemplates);
- (in editor) by ChangeStyleTemplates⁽⁵⁰⁰⁾ method;
- by RichViewActions that can modify this collection (TrvActionNew, TrvActionOpen, TrvActionInsertHyperlink, and actions calling ChangeStyleTemplates⁽⁵⁰⁰⁾: TrvActionStyleTemplates, TrvActionAddStyleTemplate).

6.1.1.4.42 TRichView.OnTextFound

Occurs when TRichView.SearchText⁽³⁴⁶⁾ (or TRichViewEdit.SearchText⁽⁵⁴³⁾) finds a text.

type

```
TRVTextFoundEvent = procedure(Sender: TCustomRichView(210);
  const AText: TRVUnicodeString(1032); ARVData: TCustomRVData(957);
  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⁽²⁴⁷⁾, a table cell⁽⁸⁹⁵⁾, 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⁽³¹²⁾ for additional information).

If **Invert** = *False*, the text should be selected from start to end. If **Invert** = *True*, 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`⁽³⁶⁷⁾ 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`⁽⁸⁰⁴⁾.`RichView`⁽⁸⁰⁷⁾
- 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(957); ItemNo: Integer;
    SaveFormat: TRVSaveFormat(1020);
    var Target, Extras: TRVUnicodeString(1032)) of object;
```

property `OnWriteHyperlink`: `TRVWriteHyperlink`;

(introduced in v1.8; changed in version 18)

By default, items tags⁽⁹¹⁾ 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`⁽²⁴⁷⁾, or table cell, or cell inplace editor's `RVData`⁽²⁴⁷⁾)

ItemNo – index of the hyperlink-item inside **RVData**.

SaveFormat can be *rsvfHTML* or *rsvfHTML* or *rsvfRTF* or *rsvfDocX*

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⁽⁹¹⁾ as a target.

```

procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString(1032));
begin
  Target := RVData.GetItemTag(302)(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(1032));
begin
  Target := RVData.GetItemTag(302)(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⁽¹⁰⁰⁵⁾ automatically.

Example 3:

```

procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString(1032));
begin
  case id-FirstJumpNo(236) 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⁽³⁸⁸⁾.

See also methods:

- SaveHTML⁽³³⁵⁾;
- SaveMarkdown⁽³⁴⁰⁾;
- SaveRTF⁽³⁴³⁾;
- SaveDocX⁽³³³⁾;
- (and methods saving these formats to streams).

See also:

- Hypertext⁽⁹²⁾;
- Saving to HTML⁽¹²⁷⁾.

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(210);
    RVData: TCustomRVData(957); ItemNo: Integer;
    SaveFormat: TRVSaveFormat(1020);
    var Props: TRVObjectExportProperties) of object;

```

property OnWriteObjectProperties: TRVWriteObjectPropertiesEvent;
(introduced in version 19)

This event occurs when saving objects of the following types:

- pictures⁽¹⁶³⁾ and hot-pictures⁽¹⁶⁶⁾;
- *bullets*⁽¹⁷⁰⁾ and *hotspots*⁽¹⁷²⁾;
- list markers⁽¹⁷⁴⁾ containing images;
- equations⁽¹⁸⁷⁾;
- 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**⁽²⁴⁷⁾, or table cell, or cell inplace editor's RVData⁽²⁴⁷⁾)

ItemNo – an index of this item inside **RVData**.

SaveFormat can be *rsvfHTML* or *rsvfRTF* or *rsvfDocX*

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
<i>rvsfRTF</i>	" (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).
<i>rvsfDocX</i>	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.
<i>rvsfHTML</i>	Occurs when saving list style and list levels. Index1 is an index of the list style in ListStyles ⁽⁶⁴⁶⁾ . 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 ⁽⁷³¹⁾	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
<i>rvsfRTF</i>	" (empty string)	Name is not saved, ignored.
<i>rvsfDocX</i>	'Pic N', where N is the initial value of Prop.Id.	Any string without line breaks.
<i>rvsfHTML</i>	" (empty string)	Name is not saved, ignored.

6.1.1.5 Classes of properties

- TRVDocParameters⁽⁴¹⁵⁾ is a type of TRichView⁽²¹⁰⁾.DocParameters⁽²³⁰⁾ property. It defines document properties like page size, margins, etc.
- TRVDocumentProperty⁽⁴²²⁾ is a type of TRichView⁽²¹⁰⁾.Document⁽²³²⁾ property. It is used for LiveBinding (in Delphi XE2+).
- TRVMarkdownProperties⁽⁴³⁹⁾ is a type of TRichView⁽²¹⁰⁾.MarkdownProperties⁽²³⁹⁾. It defines properties for reading and writing Markdown files and streams.
- TRVMarkdownDefaultItemProperties⁽⁴³⁸⁾ is a type of TRichView⁽²¹⁰⁾.MarkdownProperties⁽²³⁹⁾.DefaultItemProperties⁽⁴⁴⁴⁾. It contains default properties for objects read from Markdown files and streams.
- TRVRTFReaderProperties⁽⁴⁵⁰⁾ is a type of TRichView⁽²¹⁰⁾.RTFReadProperties⁽²⁴⁶⁾ property. It defines properties for reading RTF and DocX files and streams.

6.1.1.5.1 TRVDocParameters

This is a type of TRichView⁽²¹⁰⁾.DocParameters⁽²³⁰⁾. It contains document's and page properties that can be written to or read from RVF⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format). The components do not use these properties, but they can be assigned to TRVPrint⁽⁷⁵⁹⁾ using the method AssignDocParameters⁽⁷⁷¹⁾.

Unit [VCL/FMX] RVDocParams / fmxRVDocParams.

Syntax

```
TRVDocParameters = class(TPersistent)
```

(introduced in version 10)

Properties

This class has the following properties:

- Units⁽⁴²⁰⁾ – measurement units for margins, paper size, header and footer positions;
- PageWidth⁽⁴¹⁹⁾, PageHeight⁽⁴¹⁹⁾ – page size;
- Orientation⁽⁴¹⁸⁾ – page orientation;
- LeftMargin⁽⁴¹⁸⁾, RightMargin⁽⁴¹⁹⁾, TopMargin⁽⁴²⁰⁾, BottomMargin⁽⁴¹⁶⁾ – margins;
- MirrorMargins⁽⁴¹⁸⁾ allow to exchange left and right margins on even pages;
- HeaderY⁽⁴¹⁸⁾, FooterY⁽⁴¹⁷⁾ – position of header and footer;
- TitlePage⁽⁴¹⁹⁾ shows/hides header and footer for the first page;
- FacingPages⁽⁴¹⁷⁾ shows/hides header and footer for even pages;
- ZoomMode⁽⁴²⁰⁾ – zooming mode;
- ZoomPercent⁽⁴²¹⁾ – zooming percent.
- Author⁽⁴¹⁶⁾, Title⁽⁴¹⁹⁾, Comments⁽⁴¹⁷⁾ – text properties.

Methods

This class has the following methods:

- Reset⁽⁴²¹⁾ resets all properties to default values;
- ResetLayout⁽⁴²¹⁾ resets all sizes to default values measured in Units⁽⁴²⁰⁾;
- UnitsPerInch⁽⁴²¹⁾ allows to convert between measurement units;

- ConvertToUnits⁽⁴²¹⁾ converts all sizes to the specified units.

6.1.1.5.1.1 Properties

In TRVDocParameters

- Author⁽⁴¹⁶⁾
- BottomMargin⁽⁴¹⁶⁾
- Comments⁽⁴¹⁷⁾
- FacingPages⁽⁴¹⁷⁾
- FooterY⁽⁴¹⁷⁾
- HeaderY⁽⁴¹⁸⁾
- LeftMargin⁽⁴¹⁸⁾
- MirrorMargins⁽⁴¹⁸⁾
- Orientation⁽⁴¹⁸⁾
- PageHeight⁽⁴¹⁹⁾
- PageWidth⁽⁴¹⁹⁾
- Reset⁽⁴²¹⁾
- ResetLayout⁽⁴²¹⁾
- RightMargin⁽⁴¹⁹⁾
- Title⁽⁴¹⁹⁾
- TitlePage⁽⁴¹⁹⁾
- TopMargin⁽⁴²⁰⁾
- Units⁽⁴²⁰⁾
- UnitsPerInch⁽⁴²¹⁾
- ZoomMode⁽⁴²⁰⁾
- ZoomPercent⁽⁴²¹⁾

Author of the document

```
property Author: TRVUnicodeString(1032);
```

(introduced in version 13; changed in version 18)

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd⁽⁵⁴⁵⁾.

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⁽⁴²⁰⁾.

```
property BottomMargin: TRVLength(1015);
```

This property is not used by TRichView components, but it can be written and read from RVF⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters⁽⁷⁷¹⁾ assigns this property to TRVPrint.Margins⁽⁷⁶⁸⁾.Bottom.

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetFloatPropertyEd`⁽⁵⁴⁵⁾.

Default value:

1.0

Comments to the document (document description)

property `Comments: TRVUnicodeString`⁽¹⁰³²⁾;

(introduced in version 13; changed in version 18)

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetStrPropertyEd`⁽⁵⁴⁵⁾.

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`⁽¹²⁴⁾ (`RichView Format`), `RTF` (`Rich Text Format`) and `DocX` files.

`TRVPrint.AssignFromDocParameters`⁽⁷⁷¹⁾ assigns this property to `TRVPrint.FacingPages`⁽⁷⁶⁵⁾.

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetIntPropertyEd`⁽⁵⁴⁵⁾.

Headers and footers are assigned to `TRichView` using `SetHeader`⁽³⁵⁶⁾ and `SetFooter`⁽³⁵⁵⁾ methods.

Default value:

False

See also:

- `TitlePage`⁽⁴¹⁹⁾

Distance from the bottom of page to the bottom of footer, measured in `Units`⁽⁴²⁰⁾.

property `FooterY: TRVLength`⁽¹⁰¹⁵⁾;

This property is not used by `TRichView` components, but it can be written and read from `RVF`⁽¹²⁴⁾ (`RichView Format`) and `RTF` (`Rich Text Format`) files.

`TRVPrint.AssignFromDocParameters`⁽⁷⁷¹⁾ assigns this property to `TRVPrint.FooterY`⁽⁷⁶⁶⁾.

In `TRichViewEdit`⁽⁴⁶¹⁾, you can change this property as an editing operation, see `TRichViewEdit.SetFloatPropertyEd`⁽⁵⁴⁵⁾.

Default value:

0.5

Distance from the top of page to the top of header, measured in Units ⁽⁴²⁰⁾.

property HeaderY: TRVLength ⁽¹⁰¹⁵⁾;

This property is not used by TRichView components, but it can be written and read from RVF ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters ⁽⁷⁷¹⁾ assigns this property to TRVPrint.HeaderY ⁽⁷⁶⁷⁾.

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd ⁽⁵⁴⁵⁾.

Default value:

0.5

Left margin, measured in Units ⁽⁴²⁰⁾.

property LeftMargin: TRVLength ⁽¹⁰¹⁵⁾;

This property is not used by TRichView components, but it can be written and read from RVF ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters ⁽⁷⁷¹⁾ assigns this property to TRVPrint.Margins ⁽⁷⁶⁸⁾.Left.

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd ⁽⁵⁴⁵⁾.

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 ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters ⁽⁷⁷¹⁾ assigns this property to TRVPrint.MirrorMargins ⁽⁷⁶⁹⁾.

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd ⁽⁵⁴⁵⁾.

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 ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd ⁽⁵⁴⁵⁾.

Default value:

poPortrait

Page height, measured in Units ⁽⁴²⁰⁾.

```
property PageHeight: TRVLength(1015);
```

This property is not used by TRichView components, but it can be written and read from RVF ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

Default value:

11.692913386 (A4 height in inches)

Page width, measured in Units ⁽⁴²⁰⁾.

```
property PageWidth: TRVLength(1015);
```

This property is not used by TRichView components, but it can be written and read from RVF ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

Default value:

8.2677165354 (A4 width in inches)

Right margin, measured in Units ⁽⁴²⁰⁾.

```
property RightMargin: TRVLength(1015);
```

This property is not used by TRichView components, but it can be written and read from RVF ⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters ⁽⁷⁷¹⁾ assigns this property to TRVPrint.Margins ⁽⁷⁶⁸⁾.Right.

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd ⁽⁵⁴⁵⁾.

Default value:

1.25

Title of the document

```
property Title: TRVUnicodeString(1032);
```

(introduced in version 13; change version 18)

In TRichViewEdit ⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd ⁽⁵⁴⁵⁾.

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 ⁽¹²⁴⁾ (RichView Format), RTF (Rich Text Format) and DocX files. It also affects saving headers and footers in HTML files.

TRVPrint.AssignFromDocParameters⁽⁷⁷¹⁾ assigns this property to TRVPrint.TitlePage⁽⁷⁶⁹⁾.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

Headers and footers are assigned to TRichView using SetHeader⁽³⁵⁶⁾ and SetFooter⁽³⁵⁵⁾ methods.

Default value:

False

See also:

- FacingPages⁽⁴¹⁷⁾

Top margin, measured in Units⁽⁴²⁰⁾.

property TopMargin: TRVLength⁽¹⁰¹⁵⁾;

This property is not used by TRichView components, but it can be written and read from RVF⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters⁽⁷⁷¹⁾ assigns this property to TRVPrint.Margins⁽⁷⁶⁸⁾.Top.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd⁽⁵⁴⁵⁾.

Default value:

1.0

Measurement units for all sizes in TRVDocParameters.

property Units: TRVUnits⁽¹⁰³²⁾;

Assignment to this property does not change other properties. Use ConvertToUnits⁽⁴²¹⁾ to convert all sizes to new units.

If **Units** = *rvuPixels*, one unit = 1/Style⁽²⁵³⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾ of an inch (or 1/96, if Style⁽²⁵³⁾ = *nil*).

Use UnitsPerInch⁽⁴²¹⁾ to convert values between units.

In TRichViewEdit⁽⁴⁶¹⁾, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd⁽⁵⁴⁵⁾.

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⁽¹²⁴⁾ (RichView Format) and RTF (Rich Text Format) files.

Default value:

rvzmCustom

View scale, % (if `ZoomMode`⁴²⁰ = `rvzmCustom`).

```
property ZoomPercent: Integer;
```

This property is not used by TRichView components, but it can be written and read from RVF¹²⁴ (RichView Format), DocX, and RTF (Rich Text Format) files.

Default value:

100

6.1.1.5.1.2 Methods

In TRVDocParameters

```
ConvertToUnits421  
Reset421  
ResetLayout421  
UnitsPerInch421
```

Converts all sizes to **AUnits**.

```
procedure ConvertToUnits(AUnits: TRVUnits420);
```

This method converts all size properties from Units⁴²⁰ to **AUnits**, then assigns Units⁴²⁰ = **AUnits**.

The following properties are converted:

- PageWidth⁴¹⁹, PageHeight⁴¹⁹ – page size;
- LeftMargin⁴¹⁸, RightMargin⁴¹⁹, TopMargin⁴²⁰, BottomMargin⁴¹⁶ – margins;
- HeaderY⁴¹⁸, FooterY⁴¹⁷ – position of header and footer.

Resets all properties to default values.

```
procedure Reset;
```

Resets all sizes to default values measured in Units⁴²⁰.

```
procedure ResetLayout;
```

It makes sense to call this method on "File | New" command.

The following properties are reset:

- PageWidth⁴¹⁹, PageHeight⁴¹⁹ – page size;
- LeftMargin⁴¹⁸, RightMargin⁴¹⁹, TopMargin⁴²⁰, BottomMargin⁴¹⁶ – margins;
- HeaderY⁴¹⁸, FooterY⁴¹⁷ – position of header and footer.

In addition, this method resets Orientation⁴¹⁸ to *poPortrait* and MirrorMargins⁴¹⁸ to *False*.

Returns the number of **Units** in one inch.

```
function UnitsPerInch(Units: TRVUnits420): TRVLength1015;
```

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 ⁴²⁰.

6.1.1.5.2 TRVDocumentProperty

This is a type of TRichView ²¹⁰.Document ²³². 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 ⁵⁹⁴ and TDBRichViewEdit ⁶⁰⁶ 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 ⁴²²
- AutoDeleteUnusedStyles ⁴²³
- CodePage ⁴²³
- FieldFormat ⁴²⁴
- IgnoreEscape ⁴²⁴

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 ⁴²⁴ = *rvdbMarkdown*)
- as a plain text, otherwise

Default value:

False

See also

- CodePage ⁴²³
- TDBRichView ⁵⁹⁴.AllowMarkdown ⁵⁹⁸

- `TDBRichViewEdit`⁶⁰⁶.`AllowMarkdown`⁶¹¹

If this property is set to *True*, the component deletes unused text⁶⁵⁴, paragraph⁶⁴⁸ and list⁶⁴⁶ styles before loading.

property `AutoDeleteUnusedStyles`: `Boolean`;

This property controls loading from `TBlobField` linked to this `TRichView` using `LiveBindings` (Delphi XE2+)

If `FieldFormat`⁴²⁴ = *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`⁶³⁰ component with other `TRichView` controls (others `TRichView` controls must not be linked to the same `TRVStyle` component via `Style`²⁵³ property) and
- if the owner of this property is `TRichViewEdit`⁴⁶¹, and documents are created in this editor and saved as `RVF`¹⁰⁸, the collections of text, paragraph and list styles must be saved together with documents (see `RVFOptions`²⁴⁷, *rvfoSaveTextStyles* and *rvfoSaveParaStyles* must be included for **`AutoDeleteUnusedStyles`** = *True*).

Default value:

False

See also

- `TDBRichView`⁵⁹⁴.`AutoDeleteUnusedStyles`⁵⁹⁹
- `TDBRichViewEdit`⁶⁰⁶.`AutoDeleteUnusedStyles`⁶¹¹

Encoding for saving and loading plain text and Markdown.

property `CodePage`: `TRVCodePage`⁹⁹⁶;

(introduced in version 23)

This property controls loading and saving from/to `TBlobField` linked to/from this `TRichViewEdit`⁴⁶¹ using `LiveBindings` (Delphi XE2+).

The default value of this property is `RV_CP_DEFAULT`¹⁰⁵⁰. 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`¹⁰⁵⁰

See also

- `TDBRichView`⁵⁹⁴.`CodePage`⁵⁹⁹
- `TDBRichViewEdit`⁶⁰⁶.`CodePage`⁶¹²
- `TRichViewEdit`⁴⁶¹.`OnSaveCustomFormat`⁵⁸⁵

Format for saving data in the database field (linked using LiveBindings)

```
property FieldFormat: TRVDBFieldFormat998;
```

This property controls loading from TBlobField linked to this TRichViewEdit⁴⁶¹ 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)¹²⁴, RTF, DocX, HTML, Markdown or plain text.

Markdown and a plain text are saved using the specified CodePage⁴²³ (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 OnSaveCustomFormat⁵⁸⁵ event.

Default value:

rvdbRVF

See also

- TDBRichViewEdit⁶⁰⁶.FieldFormat⁶¹³
- TRichViewEdit⁴⁶¹.OnSaveCustomFormat⁵⁸⁵

Disallows processing of **Escape** key

```
property IgnoreEscape: Boolean;
```

This property controls loading document from TBlobField linked to this TRichViewEdit⁴⁶¹ 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:

- TDBRichViewEdit⁶⁰⁶.IgnoreEscape⁶¹⁴

6.1.1.5.3 TRVHTMLReaderProperties

This is a type of TRichView²¹⁰.HTMLReadProperties²³⁷. 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`⁴²⁷ – reading background and margins
- `ReadDocParameters`⁴²⁷ – reading title, author, and description
- `SkipHiddenText`⁴²⁸ – reading hidden content
- `IDAsCheckpoints`⁴²⁷ – reading IDs (as checkpoints⁸⁷)

Options:

- `EnforceListLevels`⁴²⁶ – choosing between keeping a list level / a list type
- `FontSizePrecision`⁴²⁶ – font size precision
- `XHTMLCheck`⁴²⁸ – turns XHTML stricter syntax checking on/off.

6.1.1.5.3.1 Properties

In TRVHtmlReaderProperties

- `BasePathLinks`⁴²⁵
- `EnforceListLevels`⁴²⁶
- `FontSizePrecision`⁴²⁶
- `IDAsCheckpoints`⁴²⁷
- `ReadBackground`⁴²⁷
- `ReadDocParameters`⁴²⁷
- `SkipHiddenText`⁴²⁸
- `XHTMLCheck`⁴²⁸

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`³⁸⁸ event occurs with `Target='download.html'` (or, if the event is not assigned, this value is stored in them item tag⁹¹);
- 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`²⁴⁶.`BasePathLinks`⁴⁵¹
- `rvmdloBasePathLinks` option in `TRichView.MarkdownProperties`²³⁹.`LoadOptions`⁴⁴⁶

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`²³⁹.`LoadOptions`⁴⁴⁶

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
<code>rvfpPoints</code>	Precision: 1 point (font sizes are loaded as integer values)
<code>rvfpHalfPoints</code>	Precision: 1/2 of a point (font sizes like 10.5 points are possible)
<code>rvfpMax</code>	VCL and LCL: the same as <code>rvfpHalfPoints</code>

Value	Meaning
	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 ⁽⁸⁷⁾.

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 ⁽²¹⁰⁾ control:

- BackgroundBitmap ⁽²²²⁾
- BackgroundStyle ⁽²²³⁾
- LeftMargin ⁽²³⁸⁾, TopMargin ⁽²⁵⁵⁾, RightMargin ⁽²⁴⁴⁾, BottomMargin ⁽²²⁵⁾

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 ⁽²¹⁰⁾.DocParameters ⁽²³⁰⁾ control:

- Title ⁽⁴¹⁹⁾ (from <title>)

- Comments⁽⁴¹⁷⁾ (from <meta name="description">'s content)
- Author⁽⁴¹⁶⁾ (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⁽¹⁰¹⁾ is added.

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

- TRVRTFReaderProperties⁽⁴⁵⁰⁾.SkipHiddenText⁽⁴⁵⁸⁾

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, stopping on errors such as unclosed tags.

This property specifies when such checking is done.

Value	Meaning
<i>rvxhtmlcYes</i>	Always
<i>rvxhtmlcNo</i>	Never
<i>rvxhtmlcAuto</i>	Only for files with "XML" and "XHTML" extensions.

Default value:*rvxhtmlcAuto*

6.1.1.5.4 TRVHTMLSaveProperties

This is a type of TRichView⁽²¹⁰⁾.HTMLSaveProperties⁽²³⁷⁾. 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⁽⁴³³⁾.

CSS can be customized using CSOptions⁽⁴³⁰⁾.

By default, CSS styles are saved as a style sheet in HTML <head>, but they can be saved inline (CSSSavingType⁽⁴³¹⁾) or externally (ExternalCSSFileName⁽⁴³²⁾).

Additional CSS styles can be provided in ExtraStyles⁽⁴³³⁾ property.

TRichView can write HTML or XHTML, depending on HTMLVersion⁽⁴³³⁾. Encoding is specified in Encoding⁽⁴³¹⁾.

Images can be saved in external files or directly in HTML, customized by ImageOptions⁽⁴³⁴⁾, ImagesPrefix⁽⁴³⁴⁾.

Checkpoints are saved as anchors, customized by UseCheckpointNames⁽⁴³⁷⁾, CheckpointsPrefix⁽⁴³⁰⁾.

List markers are saved as HTML lists or as text/images, depending on ListMarkerSavingType⁽⁴³⁶⁾.

Header and footer can be saved in HTML, if specified in SaveHeaderAndFooter⁽⁴³⁷⁾.

Note: the default value of ImageOptions⁽⁴³⁴⁾ depends on the control type:

- for TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾: []
- for TDBRichViewEdit⁽⁶⁰⁶⁾: [rvhtmlsiInlineImages]

6.1.1.5.4.1 Properties

In TRVHTMLSaveProperties

- CheckpointsPrefix⁽⁴³⁰⁾
- CSOptions⁽⁴³⁰⁾
- CSSSavingType⁽⁴³¹⁾
- DefaultOStyle⁽⁴³¹⁾
- Encoding⁽⁴³¹⁾
- ExternalCSSFileName⁽⁴³²⁾
- ExtraStyles⁽⁴³³⁾
- HTMLSavingType⁽⁴³³⁾
- HTMLVersion⁽⁴³³⁾
- ImageOptions⁽⁴³⁴⁾

- ImagesPrefix ⁽⁴³⁶⁾
- ListMarkerSavingType ⁽⁴³⁶⁾
- SaveHeaderAndFooter ⁽⁴³⁷⁾
- UseCheckpointNames ⁽⁴³⁷⁾

Specifies prefix for names of anchors in saved HTML.

```
property CheckpointsPrefix: TRVUnicodeString(1032) ;
```

(introduced in version 21)

This property is used to generate names of anchors in HTML.

It is used only if UseCheckpointNames ⁽⁴³⁷⁾ = *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
<i>rvcsssoForceNonTextCSS</i>	If set, CSS is used for saving some non-text items (such as images and tables), even if HTMLSavingType ⁽⁴³³⁾ = <i>rvhtmlstSimplified</i> .
<i>rvcsssoNoDefCSSStyle</i>	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. HTMLSavingType⁴³³ = *rvhtmlstNormal*.

Value	Meaning
<i>rvcssstStyleSheet</i>	<p>Styles are saved as a style sheet (and referred in "class" attributes of tags). Recommended.</p> <p>By default, styles are saved in <style> tag. But if ExternalCSSFileName⁴³² is defined, it is used instead.</p>
<i>rvcssstInline</i>	<p>Styles are saved inline (in "style" attributes of tags).</p> <p>This option generates highly overbloated HTML. Not recommended.</p> <p>This option may be useful when generating HTML that will be used as a part of larger HTML file.</p>

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 (TextStyles⁶⁵⁴[0] of the linked TRVStyle⁶³⁰ 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: TRVHTMLEncoding1011;
```

(introduced in version 21)

Note: for WideMemo fields, TDBRichViewEdit⁽⁶⁰⁶⁾ component stores HTML in UTF-16 encoding, ignoring **Encoding** and EncodingStr⁽⁴³²⁾.

Default value:

rvhtmlUTF8

See also:

- EncodingStr⁽⁴³²⁾

Encoding string for writing in HTML files.

```
property EncodingStr: TRVUnicodeString(1032);
```

(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⁽⁴³¹⁾ property.

Unlike Encoding⁽⁴³¹⁾, **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⁽⁶⁰⁶⁾ component stores HTML in UTF-16 encoding, ignoring **EncodingStr** and Encoding⁽⁴³¹⁾.

Example: how to save HTML in UTF-16 encoding

1. Assign Encoding⁽⁴³¹⁾ = *rvhtmlUTF8*, **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(1032);
```

(introduced in version 21)

This property is used only if HTML is saved using CSS, i.e. HTMLSavingType⁽⁴³³⁾ = *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⁽⁴³¹⁾ = *rvcssstStyleSheet*). In this case, it is assumed that this file is created using TRVStyle.SaveCSS⁽⁶⁶²⁾ method.

Specifies additional CSS styles for writing to HTML.

```
property ExtraStyles: TRVUnicodeString1032;
```

(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 HTMLSavingType⁴³³ = *rvhtmlstNormal*, and CSSSavingType⁴³¹ = *rvcssstStyleSheet*, and ExternalCSSFileName⁴³² = ""), **ExtraStyles** are written after TRichView style sheet (so **ExtraStyles** have higher priority).

ExtraStyles are always saved, even if HTMLSavingType⁴³³ = *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
<i>rvhtmlstNormal</i>	Normal HTML: formatting is represented by CSS (Cascading Style Sheets)
<i>rvhtmlstSimplified</i>	Simplified HTML: formatting is represented by deprecated HTML tags (like or). CSS is not used. For non-text items (like images or tables) CSS still can be used, if <i>rvcsssoForceNonTextCSS</i> is included in CSSOptions ⁴³⁰ .

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
<i>rvhtmlvHTML</i>	HTML 4
<i>rvhtmlvXHTML</i>	XHTML 1

Default value:*rvhtmlvHTML*

Options for saving images in HTML.

```

type // Defined in RVStyle.pas
  TRVHTMLSaveImageOption = (
    rvhtmlsioInlineImages,
    rvhtmlsioUseItemImageFileNames,
    rvhtmlsioImageSizes,
    rvhtmlsioImageSizeAttributes,
    rvhtmlsioOverrideImages
  );
  TRVHTMLSaveImageOptions =
    set of TRVHTMLSaveImageOption;

```

```

property ImageOptions: TRVHTMLSaveImageOptions;

```

(introduced in version 21, changed in v22)

Option	Meaning
<i>rvhtmlsioInlinelImages</i>	<p>If set, images are stored directly in HTML file (in base64 encoding).</p> <p>If not set, images are stored in external files.</p>
<i>rvhtmlsioUseItemImageFileNames</i>	<p>If included, images with defined (non-empty) file names will not be saved, but their file names will be written in HTML.</p> <p>File names for items are defined in <i>rvshtmlsioImageFileName</i> string property⁽¹⁰⁰⁵⁾.</p> <p>This option has higher priority than <i>rvhtmlsioInlinelImages</i>.</p> <p>See the explanation below this table.</p>
<i>rvhtmlsioImageSizes</i>	<p>If set, TRichView saves widths and heights of images in HTML explicitly, even for non-resized images.</p>

Option	Meaning
	<p>This option does not affect pictures having defined <i>rveplImageWidth</i> and/or <i>rveplImageHeight</i> item extra integer properties⁽¹⁰⁰⁰⁾. Sizes are always saved for such images.</p> <p>You can override this setting for the specific item using <i>rveNoHTMLImagesSize</i> item extra integer properties⁽¹⁰⁰⁰⁾.</p>
<i>rvhtmlsiolImageSizeAttributes</i>	<p>If set, widths and height of images are always saved as attributes of , not as CSS properties.</p> <p>Note: the current version of Microsoft Outlook does not understand image sizes specified in CSS [2023].</p>
<i>rvhtmlsioOverrideImages</i>	<p>This option is used if images are stored in external files (i.e., <i>rvhtmlsiolInlinelImages</i> is not included).</p> <p>If set, TRichView can override existing images.</p> <p>If not set, TRichView generates unique names for images.</p>

When TRichView saves an image in HTML, the following steps are performed:

1. OnHTMLSaveImage⁽³⁷⁵⁾ event occurs. If *rvhtmlsioUseItemImageFileNames* is included, *rvesplImageFileName* string property⁽¹⁰⁰⁵⁾ 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 *rvesplImageFileName* string property⁽¹⁰⁰⁵⁾ is not empty, it is written to HTML; the image file is not saved. Otherwise
3. OnSaveImage2⁽⁴⁰²⁾ occurs. If the image is not saved in this event, then
4. If *rvhtmlsiolInlinelImages* 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⁽⁴³⁶⁾ property.

On the two last steps, non-web images (such as icons or metafiles) are saved as Jpeg images.

Default value:

- for TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾: []
- for TDBRichViewEdit⁽⁶⁰⁶⁾, TcxTRichViewEdit.Properties: [*rvhtmlsiolInlinelImages*]

See also

- TRVMarkdownProperties⁽⁴³⁹⁾.SaveOptions⁽⁴⁴⁸⁾ (has similar options for saving images in Markdown)

Specifies prefix for names of image files.

```
property ImagesPrefix: TRVUnicodeString(1032);
```

(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 (*rvhtmlsiolInLineImages* is not included ImageOptions)⁽⁴³⁴⁾.

Default value:

'img'

Defines how paragraph lists⁽¹⁷⁴⁾ are saved in HTML.

```
type // Defined in RVStyle.pas
```

```
  TRVHTMLListMarkerSavingType = (
    rvhtmlmstNormal,
    rvhtmlmstAsText
  );
```

```
property ListMarkerSavingType: TRVHTMLListMarkerSavingType;
```

(introduced in version 21)

Value	Meaning
<i>rvhtmlmstNormal</i>	Normal saving, using , , tags
<i>rvhtmlmstAsText</i>	Saving list markers as text or images (depending on the marker type).

***rvhtmlmstNormal* mode**

When saving HTML using CSS (HTMLSavingType⁽⁴³³⁾ = *rvhtmlmstNormal*), CSS of lists are inserted directly in HTML (in "style" attribute):

- *rvlstBullet*, *rvlstUnicodeBullet*⁽⁷⁵⁵⁾ – if possible, exported as bullets with default markers (disc, circle or square); otherwise, as FormatString⁽⁷⁵²⁾;
- *rvlstDecimal*, *rvlstLowerAlpha*⁽⁷⁵⁵⁾ and other numbering – exported as lists with the proper numbering. FormatString⁽⁷⁵²⁾ and Font⁽⁷⁵¹⁾ are ignored;
- *rvlstPicture*, *rvlstImageList*, *rvlstImageListCounter*⁽⁷⁵⁵⁾ – exported as a bullet with picture. It's possible to change default image saving by OnHTMLSaveImage⁽³⁷⁵⁾;
- indents are saved

When saving without CSS (HTMLSavingType⁽⁴³³⁾ = *rvhtmlmstSimplified*):

- *rvlstBullet*, *rvlstUnicodeBullet*, *rvlstPicture*, *rvlstImageList*⁽⁷⁵⁵⁾ – exported as bullets with default markers (disc, circle or square). FormatString⁽⁷⁵²⁾, Picture⁽⁷⁵⁹⁾, ImageList⁽⁷⁵⁴⁾ and Font⁽⁷⁵¹⁾ are ignored;

- *rvlstDecimal*, *rvlstLowerAlpha*⁽⁷⁵⁵⁾ and other numbering – exported as lists with the proper numbering (if this numbering type cannot be represented in HTML, as a decimal numbering). *FormatString*⁽⁷⁵²⁾ and *Font*⁽⁷⁵¹⁾ are ignored;
- *rvlstImageListCounter*⁽⁷⁵⁵⁾ – saved as a decimal numbering;
- indents are not saved.

***rvhtmlmstAsText* mode**

Markers are saved without special HTML keywords for lists (like , ,).

Font⁽⁷⁵¹⁾ and format strings⁽⁷⁵²⁾ settings are respected.

When saving HTML using CSS (*HTMLSavingType*⁽⁴³³⁾ = *rvhtmlstNormal*):

LeftIndent⁽⁷⁵⁵⁾ and *MarkerIndent*⁽⁷⁵⁷⁾ are retained. *FirstIndent*⁽⁷⁵¹⁾ are retained for non-hanging markers only (*MarkerIndent*>*LeftIndent*+*FirstIndent*).

When saving without CSS (*HTMLSavingType*⁽⁴³³⁾ = *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*⁽³⁵⁶⁾/*SetFooter*⁽³⁵⁵⁾ methods.

If *DocParameters*⁽²³⁰⁾.*TitlePage*⁽⁴¹⁹⁾ = *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*⁽⁴³³⁾ = *rvhtmlstNormal*), headers and footers always use inline CSS (regardless of value of *CSSSavingType*⁽⁴³¹⁾).

Default value:

False

Specifies how checkpoints are saved to HTML.

```
property UseCheckpointNames: Boolean;
```

(introduced in version 21)

If *True*, names of checkpoints⁽⁸⁷⁾ are used as HTML anchor names. They should be unique.

If *False*, anchors names are generated using *CheckpointsPrefix*⁽⁴³⁰⁾ property.

Default value:

True

6.1.1.5.5 TRVMarkdownDefaultItemProperties

This is a type of `TRichView`⁽²¹⁰⁾.`MarkdownProperties`⁽²³⁹⁾.`DefaultItemProperties`⁽⁴⁴⁴⁾.

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`⁽⁴³⁸⁾ – color of block quotes (if `StyleTemplates` are not used)
- `BreakColor`⁽⁴³⁸⁾ – color of *breaks*⁽¹⁶⁷⁾
- `CodeBlockBackColor`⁽⁴³⁹⁾ – color of code blocks (if `StyleTemplates` are not used)

Sizes:

- `ListLeftIndentPix`, `ListMarkerIndentPix`⁽⁴³⁹⁾ – indents of list items and block quotes
- `TableCellWidthPix`⁽⁴³⁹⁾ – cell widths

Other properties:

- `ImageVAlign`⁽⁴³⁹⁾ – alignment of images

6.1.1.5.5.1 Properties

In TRVMarkdownDefaultItemProperties

- `BlockQuoteBackColor`⁽⁴³⁸⁾
- `BreakColor`⁽⁴³⁸⁾
- `CodeBlockBackColor`⁽⁴³⁹⁾
- `ImageVAlign`⁽⁴³⁹⁾
- `ListLeftIndentPix`⁽⁴³⁹⁾
- `ListMarkerIndentPix`⁽⁴³⁹⁾
- `TableCellWidthPix`⁽⁴³⁹⁾

Background color of block quote paragraphs loaded from Markdown, if `StyleTemplates` are not used.

```
property BlockQuoteBackColor: TRVColor(996);
```

This property is used only if `StyleTemplates` are not used⁽²⁵⁶⁾, or 'Quote' and 'Intense Quote' `StyleTemplates` do not exist. Otherwise, block quote paragraphs are formatted according to the `StyleTemplate`.

Default value:

`rvcNone`⁽¹⁰³⁸⁾

Color of thematic breaks⁽¹⁶⁷⁾ loaded from Markdown.

```
property BreakColor: TRVColor(996);
```

If `BreakColor = c/None`, default color is used.

Default value:*rvclNone*⁽¹⁰³⁸⁾

Background color of code block paragraphs loaded from Markdown, if StyleTemplates are not used.

property CodeBlockBackColor: TRVColor⁽⁹⁹⁶⁾;

This property is used only if StyleTemplates are not used⁽²⁵⁶⁾, or 'HTML Preformatted' StyleTemplate does not exist. Otherwise, code block paragraphs are formatted according to the StyleTemplate.

Default value:*rvclNone*⁽¹⁰³⁸⁾

Alignment of images loaded from Markdown.

property ImageVAlign: TRVVAlign⁽¹⁰³³⁾;**Default value:***rvvaBaseline*

Indent of the list marker and list text (relative to the left indent of the previous list level)

property ListMarkerIndentPix: TRVPixel96Length⁽¹⁰¹⁸⁾;**property** ListLeftIndentPix: TRVPixel96Length⁽¹⁰¹⁸⁾;

These properties are measured in pixels at 96 DPI. They are used to calculate values of MarkerIndent⁽⁷⁵⁷⁾ and LeftIndent⁽⁷⁵⁵⁾ of list levels, as well as left indents of block quotes (if StyleTemplates are not used⁽²⁵⁶⁾, or 'Quote' and 'Intense Quote' StyleTemplates do not exist; otherwise, indents specified in the StyleTemplate is used).

MarkerIndent⁽⁷⁵⁷⁾ is increased by **ListMarkerIndentPix** (converted to Units⁽⁶⁵⁵⁾).

LeftIndent⁽⁷⁵⁵⁾ is increased by **ListMarkerIndentPix** + **ListLeftIndentPix** (converted to Units⁽⁶⁵⁵⁾).

Default value:

24

Widths of cells of tables loaded from Markdown.

property TableCellWidthPix: TRVPixel96Length⁽¹⁰¹⁸⁾;

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⁽²¹⁰⁾.MarkdownProperties⁽²³⁹⁾. 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 ⁴⁴⁰:

- ImagesPrefix ⁴⁴⁵;
- LineWidth ⁴⁴⁵;
- NotesSavedAsFootnotes ⁴⁴⁸;
- SaveHiddenText ⁴⁴⁸;
- SaveOptions ⁴⁴⁸.

For importing ⁴⁴²:

- DefaultItemProperties ⁴⁴⁴;
- LoadOptions ⁴⁴⁶.

Common:

- Extensions ⁴⁴⁴.

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 ¹⁶¹ (with italic and bold attributes)
- headings
- bullets and numbering ¹⁷⁴
- horizontal lines ¹⁶⁷
- tables ¹⁷³ (optionally)
- footnotes ¹⁸⁰, endnotes ¹⁷⁸, sidenotes ¹⁸¹, text boxes ¹⁸³ (optionally, as footnotes)
- pictures
- hyperlinks

Headings

Headings are determined by OutlineLevel ⁷²⁵ 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 *rvmdsoUseSetTextHeadings* option in *SaveOptions* ⁽⁴⁴⁸⁾.

Text attributes

If *StyleTemplates* are not used ⁽²⁵⁶⁾, 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* ⁽⁷⁰⁸⁾ property)

If *StyleTemplates* are used ⁽²⁵⁶⁾, 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* ⁽⁷⁰⁸⁾ 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 ⁽²⁵⁶⁾, block quotes are not saved.

If *StyleTemplates* are used ⁽²⁵⁶⁾, 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* ⁽⁴⁴⁸⁾.

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 *rvmdTables* in *Extensions* ⁽⁴⁴⁴⁾. 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 *rvmdFootnotes* in *Extensions* ⁽⁴⁴⁴⁾, and choose note types in *NotesSavedAsFootnotes* ⁽⁴⁴⁸⁾. Otherwise, contents of notes are saved in place where they are inserted, as text inside [].

Encoding

TRichView exports Markdown in UTF-8 encoding.

If 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 (because otherwise

About Markdown import

TRichView can import from Markdown:

- text⁽¹⁶¹⁾ (with emphasis and strong emphasis, code spans)
- headings
- multilevel block quotes
- code blocks
- bullets and numbering⁽¹⁷⁴⁾
- horizontal lines⁽¹⁶⁷⁾
- tables⁽¹⁷³⁾ (optionally)
- pictures
- hyperlinks

StyleTemplates

If StyleTemplates are used⁽²⁵⁶⁾, 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⁽⁷²⁵⁾.

If StyleTemplates are used⁽²⁵⁶⁾, 'heading 1' ... 'heading 2' StyleTemplates are applied to heading paragraphs.

If StyleTemplates are not used⁽²⁵⁶⁾ (or the corresponding StyleTemplate does not exist), TRichView uses defaults for text attributes (you can modify them in the global singleton object RVDefaultLoadProperties⁽¹⁰⁵¹⁾: DefaultProperties[]⁽⁹⁶⁵⁾ 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⁽²⁵⁶⁾, '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⁽²⁵⁶⁾, these paragraphs are indented by DefaultItemProperties⁽⁴⁴⁴⁾.ListMarkerIndentPix⁽⁴³⁹⁾ and colored by DefaultItemProperties⁽⁴⁴⁴⁾.BlockQuoteBackColor⁽⁴³⁸⁾.

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⁽²⁵⁶⁾, 'HTML Preformatted' StyleTemplates is applied to paragraphs, 'HTML Code' StyleTemplate is applied to text.

If StyleTemplates are not used⁽²⁵⁶⁾, these paragraphs are colored by DefaultItemProperties⁽⁴⁴⁴⁾.CodeBlockBackColor⁽⁴³⁹⁾, and text is formatted using 'Courier New' font (you can assign another font name in RVDefaultLoadProperties⁽¹⁰⁵¹⁾.DefaultMonoSpacedFontName⁽⁹⁶⁵⁾ property).

Text attributes

If StyleTemplates are not used⁽²⁵⁶⁾, 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 'Courier New' font (you can assign another font name in RVDefaultLoadProperties⁽¹⁰⁵¹⁾.DefaultMonoSpacedFontName⁽⁹⁶⁵⁾ property)
- hyperlinks are blue and underlined (you can modify link appearance in the global singleton object RVDefaultLoadProperties⁽¹⁰⁵¹⁾: DefaultProperties[⁽⁹⁶⁵⁾rvftHyperlink] property).

If StyleTemplates are used⁽²⁵⁶⁾, 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⁽⁴⁴⁴⁾.ListMarkerIndentPix and ListLeftIndentPix⁽⁴³⁹⁾ properties.

For complex nested list, you can choose either keeping level nesting (include *rvmdloEnforceListLevels* in LoadOptions⁽⁴⁴⁶⁾) or list type (exclude *rvmdloEnforceListLevels* from LoadOptions⁽⁴⁴⁶⁾).

Tables

To enable table loading, include *rvmdTables* in Extensions⁽⁴⁴⁴⁾.

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⁽⁴⁴⁴⁾.TableCellWidthPix⁽⁴³⁹⁾.

6.1.1.5.6.1 Properties

In TRVMarkdownProperties

- DefaultItemProperties ⁽⁴⁴⁴⁾
- Extensions ⁽⁴⁴⁴⁾
- ImagesPrefix ⁽⁴⁴⁵⁾
- LineWidth ⁽⁴⁴⁵⁾
- LoadOptions ⁽⁴⁴⁶⁾
- NotesSavedAsFootnotes ⁽⁴⁴⁸⁾
- SaveHiddenText ⁽⁴⁴⁸⁾
- SaveOptions ⁽⁴⁴⁸⁾

Default properties of items used for Markdown loading

```
property DefaultItemProperties: TRVMarkdownDefaultItemProperties (438) ;  
(introduced in version 20)
```

See details in the topic about TRVMarkdownDefaultItemProperties ⁽⁴³⁸⁾.

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
<i>rvmdeTables</i>	Tables
<i>rvmdeFootnotes</i>	Footnotes

Tables extension

If support of this extension is activated, TRichView can save and load tables ⁽¹⁷³⁾ 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 `
`, if *rvmdsoUseBR* is included in SaveOptions ⁽⁴⁴⁸⁾
- 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 footnotes¹⁸⁰, endnotes¹⁷⁸, sidenotes¹⁸¹ and text boxes¹⁸³ in Markdown format as footnotes. See NotesSavedAsFootnotes⁴⁴⁸.

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: TRVUnicodeString¹⁰³²;

(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: pictures¹⁶³, hot-pictures¹⁶⁶, bullets¹⁷⁰, hotspots¹⁷².

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 SaveOptions⁴⁴⁸, 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 = (
  rvmldoBasePathLinks,
  rvmldoCollapseWhitespace,
  rvmldoCollapseWhitespaceInCodeSpans,
  rvmldoEnforceListLevels,
  rvmldoKeepLineBreaks);

TRVMarkdownLoadOptions = set of TRVMarkdownLoadOption;
```

(introduced in version 19)

Option	Meaning
<i>rvmldoBasePathLinks</i>	<p>If included, a base path is added to hyperlink targets (if they are relative paths).</p> <p>For images, a base path is added even if this option is not included (if image paths are relative paths)</p>
<i>rvmldoCollapseWhitespace</i>	<p>If included: replace all runs of whitespace characters (including spaces and tabs) by a single space (according to the Markdown specification).</p> <p>This property is applied to all text except for code spans and code blocks (in code blocks, whitespaces are loaded as they are)</p>
<i>rvmldoCollapseWhitespaceInCodeSpans</i>	<p>If included: replace all runs of whitespace characters (including spaces and tabs) by a single space in code spans (according to the Markdown specification).</p>
<i>rvmldoEnforceListLevels</i>	<p>If included: keeping list nesting levels.</p> <p>If included: keeping list type.</p>
<i>rvmldoKeepLineBreaks</i>	<p>Normally, double spaces or '\ ' at the end of lines in paragraph produce line breaks.</p> <p>If this option is included, line breaks in Markdown paragraphs produce line breaks as well.</p> <p>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)</p>

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](#) ⁽⁴⁵⁰⁾.[BasePathLinks](#) ⁽⁴⁵¹⁾, [TRVHTMLReaderProperties](#) ⁽⁴²⁴⁾.[BasePathLinks](#) ⁽⁴²⁵⁾
- [TRVHTMLReaderProperties](#) ⁽⁴²⁴⁾.[EnforceListLevels](#) ⁽⁴²⁶⁾

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
<i>rvntFootnote</i>	Footnote ⁽¹⁸⁰⁾
<i>rvntEndnote</i>	Endnote ⁽¹⁷⁸⁾
<i>rvntSidenote</i>	Sidenote ⁽¹⁸¹⁾
<i>rvntTextBox</i>	Text box ⁽¹⁸³⁾

This property is used only if *rvndeFootnotes* is included in Extensions ⁽⁴⁴⁴⁾. 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
, if *rvmdsoUseBR* is included in SaveOptions ⁽⁴⁴⁸⁾.

Default value:

[*rvntFootnote*, *rvntEndnote*, *rvntSidenote*]

Specifies whether a hidden text ⁽¹⁰¹⁾ is saved to Markdown

property SaveHiddenText: Boolean;

(introduced in version 19)

Default value:

True

Options for Markdown saving.

type // defined in RVStyle unit

```
TRVMarkdownSaveOption = (
  rvmdsoUseSetTextHeadings,
  rvmdsoCompact,
  rvmdsoListInBlockQuote,
  rvmdsoUseBR,
  rvmdsoOverrideImages,
  rvmdsoUseItemImageFileNames,
  rvmdsoInlineImages,
);
```

```
TRVMarkdownSaveOptions = set of TRVMarkdownSaveOption;
```

property SaveOptions: TRVMarkdownSaveOptions;

(introduced in version 19, updated in v21)

Option	Meaning
<i>rvmdsoUseSetextHeadings</i>	Using Setext heading (underlined paragraphs) instead of ATX headings (text surrounded by '#' characters) when possible
<i>rvmdsoCompact</i>	Storing compact (but less readable) Markdown
<i>rvmdsoListInBlockQuote</i>	This option is used for paragraphs that have both bullets/numbering and a block-quote style template ⁽⁹⁸⁾ . If not included, they are saved as block quotes in lists. If included, they are saved as lists in block quotes
<i>rvmdsoUseBR</i>	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.
<i>rvmdsoOverrideImages</i>	If set, TRichView can override existing images. If not set, TRichView generates unique names for images.
<i>rvmdsoUseItemImageFileNames</i>	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 ⁽⁴⁰²⁾ event has initial value of the Location parameter equal to the image file name. File names for items are defined via <i>rvsplImageFileName</i> string property ⁽¹⁰⁰⁵⁾ .
<i>rvmdsoInlineImages</i>	If included, images are stored directly in Markdown text (in base64 encoding). Otherwise images are saved in external files.

Default value:

- for TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾:
[*rvmdsoUseSetextHeadings*, *rvmdsoUseItemImageFileNames*]
- for TDBRichViewEdit⁽⁶⁰⁶⁾, TcxTRichViewEdit.Properties:
[*rvmdsoUseSetextHeadings*, *rvmdsoUseItemImageFileNames*, *rvmdsoInlineImages*]

See also

- TRVHTMLSaveProperties⁽⁴²⁹⁾.ImageOptions⁽⁴³⁴⁾ (has similar options for saving images in HTML)

6.1.1.5.7 TRVRTFReaderProperties

This is a type of TRichView⁽²¹⁰⁾.RTFReadProperties⁽²⁴⁶⁾. 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⁽⁴⁵⁶⁾;
- TextStyleMode⁽⁴⁵⁸⁾.

The following properties allow to ignore some part of RTF content:

- IgnoreBookmarks⁽⁴⁵⁴⁾;
- IgnoreFields⁽⁴⁵⁵⁾;
- IgnoreNotes⁽⁴⁵⁵⁾;
- IgnorePictures⁽⁴⁵⁵⁾;
- IgnoreSequences⁽⁴⁵⁵⁾;
- IgnoreTables⁽⁴⁵⁵⁾;
- SkipHiddenText⁽⁴⁵⁸⁾;
- ReadDocParameters⁽⁴⁵⁷⁾.

Reading Hyperlinks

By default, hyperlinks targets are loaded in items tags⁽⁹¹⁾. Use TCustomRichView.OnReadHyperlink⁽³⁸⁸⁾ to customize loading.

Reading Images

The following types of RTF pictures are supported (if IgnorePictures⁽⁴⁵⁵⁾ = *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⁽⁴⁵⁴⁾;
- 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⁽¹⁰⁵⁷⁾.RegisterPngGraphic);
- external images, loaded from files (TCustomRichView.OnImportPicture⁽³⁷⁸⁾ event occurs).

DocX may contain images of other types (such as Gif).

6.1.1.5.7.1 Properties

In TRVRTFReaderProperties

- AutoHideTableGridLines⁽⁴⁵¹⁾
- BasePathLinks⁽⁴⁵¹⁾
- CanReuseNumberedLists⁽⁴⁵²⁾

- CharsetForUnicode ⁽⁴⁵²⁾
- ConvertHighlight ⁽⁴⁵²⁾
- ConvertSymbolFonts ⁽⁴⁵³⁾
- ▶ DocXErrorCode ⁽⁴⁵⁴⁾
- ExtractMetafileBitmaps ⁽⁴⁵⁴⁾
- IgnoreBookmarks ⁽⁴⁵⁴⁾
- IgnoreFields ⁽⁴⁵⁵⁾
- IgnoreNotes ⁽⁴⁵⁵⁾
- IgnorePictures ⁽⁴⁵⁵⁾
- IgnoreSequences ⁽⁴⁵⁵⁾
- IgnoreTables ⁽⁴⁵⁵⁾
- LineBreaksAsParagraphs ⁽⁴⁵⁶⁾
- ParaStyleMode ⁽⁴⁵⁶⁾
- ParaStyleNo ⁽⁴⁵⁷⁾
- ReadDocParameters ⁽⁴⁵⁷⁾
- ReadNoteNumberingType ⁽⁴⁵⁷⁾
- ▶ RTFErrorCode ⁽⁴⁵⁷⁾
- SkipHiddenText ⁽⁴⁵⁸⁾
- TextStyleMode ⁽⁴⁵⁸⁾
- TextStyleNo ⁽⁴⁵⁹⁾
- UseCharsetForUnicode ⁽⁴⁵⁹⁾
- UseHypertextStyles ⁽⁴⁵⁹⁾
- UseSingleCellPadding ⁽⁴⁶⁰⁾

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 ⁽⁸⁶¹⁾ 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 *rvShowGridLines* option in TRichView.Options ⁽²⁴⁰⁾.

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 ⁽³⁸⁸⁾ event occurs with Target='MyFile2.rtf' (or, if the event is not assigned, this value is stored in them item tag ⁽⁹¹⁾);

- 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⁽²³⁷⁾.BasePathLinks⁽⁴²⁵⁾
- rvmldoBasePathLinks option in TRichView.MarkdownProperties⁽²³⁹⁾.LoadOptions⁽⁴⁴⁶⁾

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⁽⁷⁰⁰⁾ for Unicode text read from RTF and DocX.

property CharsetForUnicode: TRVFontCharset⁽¹⁰⁰⁸⁾;

(introduced in version 11)

This property is used if UseCharsetForUnicode⁽⁴⁵⁹⁾ = *True* and TextStyleMode⁽⁴⁵⁸⁾ <> *rtrsUseSpecified*.

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
<i>rtfhlIgnore</i>	Highlighting is ignored.
<i>rtfhlFixedColors</i>	Highlighting is converted to text background using a fixed set of colors (according to the old RTF specification)
<i>rtfhlColorTable</i>	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 RVDefaultLoadProperties¹⁰⁵¹.DefaultUnicodeSymbolFontName⁹⁶⁵. The default value for this font name depends on the platform.

Default value:

- VCL and LCL: *False*
- FireMonkey: *True*

Contains error code for the last DocX reading operation.

```
ErrorCode: TRVRTFErrorCode;
```

```
type // defined in RVDocXReader unit.
```

```
TRVDocXErrorCode = (
  rvdockxreOK,           // No error
  rvdockxreUnpack       // File unpacking error (maybe not a DocX file)
  rvdockxreStyleSheet   // Error while parsing a style sheet
  rvdockxreDocument,   // Error while parsing the main document
  rvdockxreAssertion,  // Assertion failure (internal parser error)
  rvdockxreException,  // Exception (internal parser error)
  rvdockxreFileOpenError // Parser cannot open input file
);
```

See also:

- RTFErrorCode ⁴⁵⁷

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* ⁸⁷. Bookmark ends cannot be read.

Default value:

False

Disallows loading "page number"¹⁸⁵ and "page count"¹⁸⁶ 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 footnotes¹⁸⁰, endnotes¹⁷⁸ and references to them¹⁸⁴ 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:

- Style²⁵³.FootnotePageReset⁶⁴⁰;
- Style²⁵³.FootnoteNumbering and EndnoteNumbering⁶³⁷ (if ReadNoteNumberingType⁴⁵⁷ = *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 ExtractMetafileBitmaps⁴⁵⁴ property);
- Enhanced Metafiles, loaded as TMetafile,
- JPEG (D3+), loaded as TjpegImage,
- PNG, see RVGraphicHandler¹⁰⁵⁷.RegisterPngGraphic.

Default value:

False

Disallows loading numbering sequences¹⁷⁷ 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: TRVReaderStyleMode¹⁰¹⁹;

This property defines how RTF/DocX paragraph attributes will be converted to TRichView.Style²⁵³.ParaStyles⁶⁴⁸.

Mode	Meaning
<i>rversUseSpecified</i>	All paragraph formatting in RTF/DocX will be ignored, all paragraphs will have style specified in ParaStyleNo ⁴⁵⁷ property.
<i>rversUseClosest</i>	RTF/DocX formatting will be mapped to the the most similar existing paragraph style ⁶⁴⁸
<i>rversAddIfNeeded</i>	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 ⁶⁴⁸ and use it.

List markers (bullets and numbering) are imported only if ParaStyleMode=*rversAddIfNeeded*.

If the both TextStyleMode⁴⁵⁸=**ParaStyleMode**=*rversAddIfNeeded*, and TRichView.UseStyleTemplates²⁵⁶=*True*, and StyleTemplateInsertMode²⁵⁴<>*rvstimIgnoreSourceStyleTemplates*, the RTF style sheet will be read in TRichView.Style²⁵³.StyleTemplates⁶⁵². 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.StyleTemplateInsertMode²⁵⁴.

Default value:

- *rversUseClosest* for VCL/LCL components created at runtime.
- *rversAddIfNeeded* for FireMonkey components created at runtime.

- For components placed on form at designtime, initial value of this property is defined in the component editor⁽²¹⁸⁾ 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⁽⁴⁵⁷⁾.
- TextStyleMode⁽⁴⁵⁸⁾.

Default paragraph style for RTF and DocX loading.

property ParaStyleNo: Integer

This property is used if ParaStyleMode⁽⁴⁵⁶⁾ = *rvrsUseSpecified*.

It is an index in the collection of paragraph styles⁽⁶⁴⁸⁾.

Default value:

0

See also:

- ParaStyleMode⁽⁴⁵⁶⁾;
- TextStyleNo⁽⁴⁵⁹⁾.

Set to True to allow reading DocParameters⁽²³⁰⁾ from RTF and DocX.

property ReadDocParameters: Boolean;

(introduced in version 10)

Default value:

False

Set to True to allow reading Style⁽²⁵³⁾.FootnoteNumbering and EndnoteNumbering⁽⁶³⁷⁾ from RTF and DocX.

property ReadNoteNumberingType: Boolean;

(introduced in version 22)

This property is used only if IgnoreNotes⁽⁴⁵⁵⁾ = *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⁽⁴⁵⁴⁾

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⁽¹⁰¹⁾ is added.

Default value:

True

See also:

- TRVHtmlReaderProperties⁽⁴²⁴⁾.SkipHiddenText⁽⁴²⁸⁾

Defines a mode for extraction of character formatting from RTF and DocX

property TextStyleMode: TRVReaderStyleMode⁽¹⁰¹⁹⁾;

This property defines how RTF/DocX text attributes will be converted to TRichView.Style⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾.

Mode	Meaning
<i>rversUseSpecified</i>	All character formatting in RTF/DocX will be ignored, all text will have style specified in TextStyleNo ⁽⁴⁵⁹⁾ property.
<i>rversUseClosest</i>	RTF/DocX formatting will be mapped to the the most similar existing text style ⁽⁶⁵⁴⁾ .
<i>rversAddIfNeeded</i>	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 ⁽⁶⁵⁴⁾ and use it.

If the both **TextStyleMode**=ParaStyleMode⁽⁴⁵⁶⁾=*rversAddIfNeeded*, and TRichView.UseStyleTemplates⁽²⁵⁶⁾=*True*, and StyleTemplateInsertMode⁽²⁵⁴⁾<>*rvstimIgnoreSourceStyleTemplates*, the RTF/DocX style sheet will be read in TRichView.Style⁽²⁵³⁾

.StyleTemplates⁽⁶⁵²⁾. 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⁽²⁵⁴⁾.

Default value:

- *rversUseClosest* for VCL/LCL components created at runtime.
- *rversAddIfNeeded* for FireMonkey components created at runtime.
- For components placed on form at design time, initial value of this property is defined in the component editor⁽²¹⁸⁾ for TRichView. By default, the component editor applies "allow adding styles dynamically" mode to new TRichView components, and this property is set to *rversAddIfNeeded*.

See also:

- TextStyleNo⁽⁴⁵⁹⁾;
- ParaStyleMode⁽⁴⁵⁶⁾.

Default text style for RTF and DocX loading.

property TextStyleNo: Integer

This property is used if TextStyleMode⁽⁴⁵⁸⁾ = *rversUseSpecified*.

It is an index in the collection of text styles⁽⁶⁵⁴⁾.

Default value:

0

See also:

- TextStyleMode⁽⁴⁵⁸⁾;
- ParaStyleNo⁽⁴⁵⁷⁾.

Obsolete property, it does nothing

property UnicodeMode: TRVReaderUnicode;

type

```
TRVReaderUnicode =
  (rvruMixed, rvruNoUnicode, rvruOnlyUnicode);
```

Specifies whether CharSetForUnicode⁽⁴⁵²⁾ 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⁽⁴⁵⁸⁾ = *rversUseClosest* or *rversAddIfNeeded*).

Note: it's not recommended to change value of this property.

Default value:*False***See also:**

- Hypertext in TRichView⁽⁹²⁾.

Set it to *True* if you want to use the same values for CellHPadding⁽⁸⁵⁵⁾ and CellVPadding⁽⁸⁵⁷⁾ 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⁽¹²⁴⁾ 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⁽⁴⁶⁰⁾

6.1.1.6.1 Loading UTF-8 files

This example shows how to load Unicode UTF-8 file:

```
procedure LoadUTF8(rv: TCustomRichView(210); const FileName: String;  
  StyleNo, ParaNo: Integer);  
var Stream: TFileStream;  
    s: TRVRawByteString(1019);  
    ws: TRVUnicodeString(1032);  
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(274)(ws, StyleNo, ParaNo, ParaNo, False);  
end;
```

Call:

```
LoadUTF8(RichViewEdit1, 'test.txt', 0, 0);  
RichViewEdit1.Format(288);
```

See also:

- Unicode in TRichView⁽¹³⁰⁾.

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(TCustomRichView210)
TRichViewEdit = class(TCustomRichViewEdit)
```

RichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style²⁵³ property.

The most important property settings can be done in the component editor²¹⁸ 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:

- OnCheckpointVisible³⁷² event;
- all features related to hypertext ids: OnRVMouseMove³⁹⁵, OnJump³⁸², GetJumpPointLocation³⁰⁴ work only when user presses and holds **Ctrl** key, if *rvtoCtrlJumps* in EditorOptions⁴⁷⁵ (or if *ReadOnly*⁴⁸⁰ = True).

Hierarchy

► Hierarchy (VCL and LCL)

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRVControl
*TRVScroller*⁸¹³
*TCustomRichView*²¹⁰
TCustomRichViewEdit

► Hierarchy (FireMonkey)

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBar
TCustomRVControl
*TRVScroller*⁸¹³
*TCustomRichView*²¹⁰
TCustomRichViewEdit

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

<ul style="list-style-type: none"> ↳ Properties ⁴⁶² ↳ Events ⁴⁶² ↳ General methods ⁴⁶³ ↳ Inserting items at the position of caret ⁴⁶³ ↳ Clipboard operations ⁴⁶³ ↳ Advanced methods for modifying items ⁴⁶⁴ ↳ Information about item at the position of caret ⁴⁶⁴ ↳ Modifying item at the position of caret ⁴⁶⁵ 	<ul style="list-style-type: none"> ↳ Style templates ⁴⁶⁵ ↳ Undo/Redo ⁴⁶⁵ ↳ Unicode ⁴⁶⁶ ↳ Live spelling check ⁴⁶⁶ ↳ "Smart Popups" ⁴⁶⁶ ↳ Related to tables and special items ⁴⁶⁶ ↳ Working with <i>checkpoints</i> ⁴⁶⁶ ↳ LiveBindings and DB ⁴⁶⁷
---	---

Properties

- `CurlItemNo` ⁴⁷² – index of item at the position of caret, `OffsetInCurlItem` ⁴⁷⁹ – offset of caret in this item;
- `CurlItemStyle` ⁴⁷³ – style (or type) of the item at the position of caret;
- `CurParaStyleNo` ⁴⁷³ – current paragraph style index;
- `CurTextStyleNo` ⁴⁷⁴ – current text style index, see also `ActualCurTextStyleNo` ⁴⁷²;
- `EditorOptions` ⁴⁷⁵ – options for editing;
- `ReadOnly` ⁴⁸⁰ disables/enables editing;
- `Modified` ⁴⁷⁹ – all editing methods set it to *True*.
- `CustomCaretInterval` ⁴⁷⁴ – a blinking interval for a custom caret.
- `AcceptDragDropFormats`, `AcceptPasteFormats` ⁴⁷⁰ – lists of formats that the editor can accept.
- `DefaultPictureVAlign` ⁴⁷⁵ – alignment for pasted and dropped pictures.
- `ForceFieldHighlight` ⁴⁷⁸ affects highlighting of label items ¹⁷⁶.
- `KeyboardType` ⁴⁷⁸ [FMX] – the type of the virtual keyboard

See also properties for undo/redo.

Events

- `OnChange` ⁵⁷² occurs when the document is changed;
- `OnCheckStickingItems` ⁵⁷³ allows to add additional text protection;
- `OnCurParaStyleChanged` ⁵⁷³ occurs when the index of the current paragraph style is changed;
- `OnCurTextStyleChanged` ⁵⁷⁴ occurs when the index of the current text style is changed;
- `OnPaste` ⁵⁸⁴ allows pasting data from the Clipboard in custom formats;
- `OnStyleConversion` ⁵⁸⁵ occurs while executing `ApplyStyleConversion` ⁴⁹²;
- `OnParaStyleConversion` ⁵⁸² occurs while executing `ApplyParaStyleConversion` ⁴⁹¹;
- `OnCaretGetOut` ⁵⁷¹ occurs when the user moves the caret "outside" the editor;
- `OnCaretMove` ⁵⁷² occurs when the caret is moved;
- `OnItemTextEdit` ⁵⁷⁸ occurs when a text item ¹⁶¹'s text is changed as a result of editing operation;
- `OnSmartPopupClick` ⁵⁸⁵ occurs when user clicks a "smart popup" ⁴⁶⁶ button.
- `OnDrawCurrentLine` ⁵⁷⁴ allows to highlight the current line
- `OnMeasureCustomCaret` ⁵⁷⁸ and `OnDrawCustomCaret` ⁵⁷⁵ allow drawing your own caret instead of the system one.
- `OnOleDragEnter` ⁵⁷⁹, `OnOleDragOver` ⁵⁸⁰, `OnOleDragLeave` ⁵⁷⁹, `OnOleDrop` ⁵⁸¹, `OnBeforeOleDrop` and `OnAfterOleDrop` ⁵⁷⁰ occur on OLE drag&drop.

General Methods

- `ApplyParaStyle`⁽⁴⁹⁰⁾, `ApplyTextStyle`⁽⁴⁹⁴⁾ apply paragraph and text style to the selection;
- `ApplyStyleConversion`⁽⁴⁹²⁾ applies custom style conversion to the selected text;
- `ApplyParaStyleConversion`⁽⁴⁹¹⁾ applies custom paragraph style conversion to the selected paragraphs;
- `Change`⁽⁴⁹⁹⁾ generates `OnChange`⁽⁵⁷²⁾ event;
- `BeginUpdate`⁽⁴⁹⁸⁾, `EndUpdate`⁽⁵⁰³⁾ disable and enable redrawing;
- `SearchText`, `-A`, `-W`⁽⁵⁴³⁾ search for substring;
- `MoveCaret`⁽⁵³⁴⁾ moves the caret in the specified direction.

Inserting Items at the position of Caret (Insert*** methods)

- `InsertText`, `-A`, `-W`⁽⁵³²⁾, `InsertStringTag`, `InsertStringATag`, `InsertStringWTag`⁽⁵³⁰⁾ insert text;
- `InsertBreak`⁽⁵¹⁵⁾ inserts *break* (horizontal line);
- `InsertBullet`⁽⁵¹⁵⁾ inserts *bullet*;
- `InsertControl`⁽⁵¹⁷⁾ inserts Delphi control;
- `InsertHotspot`⁽⁵²⁰⁾ inserts *hotspot*;
- `InsertPicture`⁽⁵²⁶⁾ inserts picture;
- `InsertHotPicture`⁽⁵¹⁹⁾ inserts picture-hypertext link;
- `InsertTab`⁽⁵³¹⁾ inserts tabulator;
- `InsertRVFFromFileEd`⁽⁵²⁸⁾, `InsertRVFFromStreamEd`⁽⁵²⁹⁾ insert RVF (RichView Format) from a file or a stream;
- `InsertHTMLFromFileEd`⁽⁵²¹⁾, `InsertHTMLFromStreamEd`⁽⁵²¹⁾ insert HTML from a file or a stream;
- `InsertRTFFromFileEd`⁽⁵²⁷⁾, `InsertRTFFromStreamEd`⁽⁵²⁸⁾ insert RTF (Rich Text Format) from a file or a stream;
- `InsertTextFromFile`⁽⁵³³⁾, `InsertOEMTextFromFile`⁽⁵²⁵⁾, `InsertTextFromFileW`⁽⁵³³⁾ insert text from a file;
- `InsertMarkdownFromFileEd`⁽⁵²³⁾, `InsertMarkdownFromStreamEd`⁽⁵²⁴⁾ insert Markdown from a file or a stream;
- `InsertItem`⁽⁵²²⁾ – general method for inserting items, inserts one item (usually used for tables).

Clipboard Operations

Main methods and events:

- `CanPaste`⁽⁴⁹⁸⁾ – "can something be pasted from the Clipboard?";
- `CanPasteRVF`⁽⁴⁹⁹⁾ – "can RVF (RichView Format) be pasted from the Clipboard?";
- `CanPasteHTML`⁽⁴⁹⁹⁾ – "can HTML be pasted from the Clipboard?";
- `CanPasteRTF`⁽⁴⁹⁹⁾ – "can RTF (Rich Text Format) be pasted from the Clipboard?";
- `CutDef`⁽⁵⁰²⁾ cuts selection to the Clipboard;
- `Paste`⁽⁵³⁴⁾ pastes from the Clipboard;
- `PasteBitmap`⁽⁵³⁵⁾ pastes bitmap from the Clipboard;
- `PasteMetafile`⁽⁵³⁷⁾ pastes metafile from the Clipboard;
- `PasteGraphicFile`⁽⁵³⁵⁾ pastes metafile from the Clipboard;
- `PasteRVF`⁽⁵³⁸⁾ pastes RVF (RichView Format) from the Clipboard;
- `PasteHTML`⁽⁵³⁶⁾ pastes RTF (Rich Text Format) from the Clipboard;
- `PasteRTF`⁽⁵³⁷⁾ pastes RTF (Rich Text Format) from the Clipboard;
- `PasteTextA`⁽⁵³⁸⁾ pastes ANSI text from the Clipboard;
- `PasteTextW`⁽⁵³⁸⁾ pastes Unicode text from the Clipboard;
- event `OnPaste`⁽⁵⁸⁴⁾ allows to paste data from the Clipboard in custom formats.

Also:

- DeleteSelection⁽⁵⁰³⁾ deletes selection;
- SelectCurrentWord⁽⁵⁴⁵⁾ selects word at the position of caret;
- SelectCurrentLine⁽⁵⁴⁵⁾ selects the line containing caret.

Advanced Methods for Modifying Items (Set*InfoEd methods)****Main methods:**

- SetBreakInfoEd⁽⁵⁴⁷⁾ – for changing *break*;
- SetBulletInfoEd⁽⁵⁴⁸⁾ – for changing *bullet*;
- SetControllInfoEd⁽⁵⁵⁰⁾ – for changing inserted control;
- SetHotspotInfoEd⁽⁵⁶⁰⁾ – for changing *hotspot*;
- SetPictureInfoEd⁽⁵⁶⁵⁾ – for changing picture or *hot picture*.

Also:

- SetItemExtraIntProperty[Ex]Ed⁽⁵⁶²⁾,
SetItemExtraStrProperty[Ex]Ed⁽⁵⁶²⁾ – for changing additional properties of items;
- SetItemTagEd⁽⁵⁶³⁾ – for changing *tag*;
- SetItemVAlignEd⁽⁵⁶⁵⁾ – for changing item position relative to the line;
- SetItemTextEd⁽⁵⁶⁴⁾ changes text (name) of item;
- SetCheckpointInfoEd⁽⁵⁴⁹⁾ – for changing *checkpoint*;
- RemoveCheckpointEd⁽⁵⁴¹⁾ – for deleting *checkpoint*;
- ResizeControl⁽⁵⁴²⁾ – for resizing inserted control;
- AdjustControlPlacement⁽⁴⁸⁹⁾ returns control to the proper position;
- AdjustControlPlacement2⁽⁴⁸⁹⁾ 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⁽⁴⁷²⁾, ...) but work also in inplace editors of tables⁽¹⁹⁰⁾.

Main methods:

- GetCurrentTextInfo⁽⁵¹⁴⁾ – about text item;
- GetCurrentBreakInfo⁽⁵⁰⁴⁾ – about *break*;
- GetCurrentBulletInfo⁽⁵⁰⁵⁾ – about *bullet*;
- GetCurrentControlInfo⁽⁵⁰⁶⁾ – about inserted control;
- GetCurrentHotspotInfo⁽⁵⁰⁷⁾ – about *hotspot*;
- GetCurrentPictureInfo⁽⁵¹³⁾ – about picture or *hot picture*;
- GetCurrentItem⁽⁵⁰⁸⁾ – as an object about any item (usually for tables⁽¹⁹⁰⁾);
- GetCurrentItemEx⁽⁵⁰⁹⁾.

Also:

- GetCurrentItemExtraIntProperty[Ex]⁽⁵¹⁰⁾,
GetCurrentItemExtraStrProperty[Ex]⁽⁵¹⁰⁾ return value of additional item property;
- GetCurrentTag⁽⁵¹⁴⁾ returns item's *tag*;
- GetCurrentItemVAlign⁽⁵¹²⁾ returns item position relative to the line;
- GetCurrentCheckpoint⁽⁵⁰⁶⁾ returns item's *checkpoint*;
- GetCurrentItemText⁽⁵¹¹⁾ returns text/name of item;
- GetCurrentLineCol⁽⁵¹²⁾ 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⁽⁴⁷²⁾, ...) but work also in inplace editors of tables⁽¹⁹⁰⁾.

Main methods:

- SetCurrentBreakInfo⁽⁵⁵¹⁾ modifies *break*;
- SetCurrentBulletInfo⁽⁵⁵²⁾ modifies *bullet*;
- SetCurrentHotspotInfo⁽⁵⁵⁵⁾ modifies *hotspot*;
- SetCurrentControlInfo⁽⁵⁵⁴⁾ modifies inserted control;
- SetCurrentPictureInfo⁽⁵⁵⁹⁾ modifies picture or *hot picture*.

Also:

- SetCurrentItemExtraIntProperty[Ex]⁽⁵⁵⁶⁾,
- SetCurrentItemExtraStrProperty[Ex]⁽⁵⁶²⁾ change value of additional item property;
- SetCurrentTag⁽⁵⁶⁰⁾ modifies item's *tag*;
- SetCurrentItemVAlign⁽⁵⁵⁸⁾ changes item position relative to the line;
- SetCurrentItemText⁽⁵⁵⁷⁾ changes text (name) of item;
- SetCurrentCheckpointInfo⁽⁵⁵³⁾ modifies *checkpoint*;
- RemoveCurrentCheckpoint⁽⁵⁴¹⁾ deletes *checkpoint*;
- ResizeCurrentControl⁽⁵⁴³⁾ returns control to the proper position in document;
- RemoveCurrentPageBreak⁽⁵⁴²⁾ removes "explicit page break before item" flag;
- InsertPageBreak⁽⁵²⁵⁾ sets "explicit page break before item" flag; can split current item into two parts.

Style templates

Methods:

- ApplyStyleTemplate⁽⁴⁹³⁾, ApplyTextStyleTemplate⁽⁴⁹⁴⁾, ApplyParaStyleTemplate⁽⁴⁹¹⁾ apply the specified style template to the selection;
- ChangeStyleTemplates⁽⁵⁰⁰⁾ allows changing the collection of style templates as an editing operation.

Undo/Redo

Main:

- property UndoLimit⁽⁴⁸¹⁾ sets capacity of undo buffer;
- UndoAction⁽⁵⁶⁷⁾ returns which operation will be undone next;
- Undo⁽⁵⁶⁷⁾ undoes the last operation;
- UndoName⁽⁵⁶⁸⁾ returns name of custom undo operation which will be undone next;
- RedoAction⁽⁵⁴⁰⁾ returns which operation will be redone next;
- Redo⁽⁵⁴⁰⁾ redoes the last undone operation;
- RedoName⁽⁵⁴⁰⁾ returns name of custom redo operation which will be redone next;
- ClearUndo⁽⁵⁰¹⁾ clears undo and redo buffers.

Undo grouping:

- BeginUndoGroup⁽⁴⁹⁷⁾,
- BeginUndoCustomGroup⁽⁴⁹⁶⁾,
- SetUndoGroupMode⁽⁵⁶⁷⁾.

Alternative undo grouping:

- BeginUndoGroup2, EndUndoGroup2⁽⁴⁹⁷⁾;
- BeginUndoCustomGroup2, EndUndoCustomGroup2⁽⁴⁹⁶⁾.

Unicode

- PasteTextW⁽⁵³⁸⁾ pastes Unicode text from the Clipboard;
- InsertTextW⁽⁵³²⁾ inserts Unicode text;
- InsertTextFromFileW⁽⁵³³⁾ inserts Unicode text from file.

Live Spelling Check

- GetCurrentMisspelling⁽⁵¹³⁾ returns misspelled word at the caret position;
- [FMX] AddSpellingMenuItems⁽⁴⁸⁸⁾ adds spelling check suggestions to a popup menu;
- property LiveSpellingMode⁽⁴⁷⁹⁾ controls when live spelling thread starts;
- [FMX] event OnGetSpellingSuggestions⁽³⁷⁵⁾ 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⁽⁴⁸⁰⁾ – popup button properties;
- SmartPopupVisible⁽⁴⁸¹⁾ – popup button properties.

Events:

- OnSmartPopupClick⁽⁵⁸⁵⁾ occurs when user clicks "smart popup" button.

Related to Tables⁽⁸⁴⁶⁾ and Special Items

- InsertItem⁽⁵²²⁾ – general method for inserting items, inserts one item (usually used for tables⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, references to notes⁽¹⁸⁴⁾).
- GetCurrentItem⁽⁵⁰⁸⁾ returns item at the position of caret as an object (usually used for tables⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, references to notes⁽¹⁸⁴⁾);
- BeginItemModify⁽⁴⁹⁵⁾, EndItemModify⁽⁵⁰³⁾ are used to reformat editor after operations on item (usually on tables);
- CanChange⁽⁴⁹⁸⁾ checks possibility of performing editing operation (usually it is called before making operations on table).

Working with Checkpoints⁽⁸⁷⁾

Editing-style analogs of methods of TRichView⁽²¹⁰⁾:

- SetCheckpointInfoEd⁽⁵⁴⁹⁾ – adds *checkpoint* for the specified item;
- RemoveCheckpointEd⁽⁵⁴¹⁾ – deletes *checkpoint* for the specified item.

Methods working with the item at the position of caret:

- GetCurrentCheckpoint⁽⁵⁰⁶⁾ – returns *checkpoint* associated with the item at the caret position;
- SetCurrentCheckpointInfo⁽⁵⁵³⁾ – adds/modifies *checkpoint* for the item at the caret position;
- RemoveCurrentCheckpoint⁽⁵⁴¹⁾ – deletes *checkpoint* for the item at the caret position.

Methods working with the checkpoint exactly at the position of caret:

- InsertCheckpoint⁵¹⁶ – inserts *checkpoint* in the caret position;
- GetCheckpointAtCaret⁵⁰³ – returns the *checkpoint* exactly at the caret position;
- RemoveCheckpointAtCaret⁵⁴¹ – deletes the *checkpoint* exactly at the caret position.

LiveBindings and Databases

Events:

- OnSaveCustomFormat⁵⁸⁵ allows saving data in DB field in custom formats

6.1.2.1 Properties

Derived from TCustomRichViewEdit

- AcceptDragDropFormats⁴⁷⁰
- AcceptPasteFormats⁴⁷⁰
- ▶ ActualCurTextStyleNo⁴⁷²
- ▶ CurlItemNo⁴⁷²
- ▶ CurlItemStyle⁴⁷³
- CurParaStyleNo⁴⁷³
- CurTextStyleNo⁴⁷⁴
- CustomCaretInterval⁴⁷⁴
- DefaultPictureVAlign⁴⁷⁵
- EditorOptions⁴⁷⁵
- ForceFieldHighlight⁴⁷⁸
- KeyboardType⁴⁷⁸ [FMX]
- LiveSpellingMode⁴⁷⁹
- Modified⁴⁷⁹
- ▶ OffsetInCurlItem⁴⁷⁹
- ReadOnly⁴⁸⁰
- SmartPopupProperties⁴⁸⁰
- SmartPopupVisible⁴⁸¹
- ▶ TopLevelEditor⁴⁸¹
- UndoLimit⁴⁸¹

Derived from TCustomRichView²¹⁰

- AllowSelection²²² (deprecated)
- AnimationMode²²²
- BackgroundBitmap²²²
- BackgroundStyle²²³
- BiDiMode²²⁴
- BottomMargin²²⁵
- ClearLeft²²⁵
- ClearRight²²⁶
- Color²²⁶
- CPEventKind²²⁷
- Cursor²²⁷
- DarkMode²²⁸
- Delimiters²²⁸
- DocObjects²²⁹

- DocParameters ⁽²³⁰⁾
- DocProperties ⁽²³¹⁾
- Document ⁽²³²⁾
- ▶ DocumentHeight ⁽²³³⁾
- DocumentPixelsPerInch ⁽²³³⁾ [VCL,LCL]
- DoInPaletteMode ⁽²³⁴⁾
- ▶ FirstItemVisible ⁽²³⁶⁾
- FirstJumpNo ⁽²³⁶⁾
- HTMLReadProperties ⁽²³⁷⁾
- HTMLSaveProperties ⁽²³⁷⁾
- ▶ ItemCount ⁽²³⁷⁾
- ▶ LastItemVisible ⁽²³⁸⁾
- LeftMargin ⁽²³⁸⁾
- MarkdownProperties ⁽²³⁹⁾
- MaxLength ⁽²³⁹⁾
- MaxTextWidth ⁽²³⁹⁾
- MinTextWidth ⁽²⁴⁰⁾
- NoteText ⁽²⁴⁰⁾
- Options ⁽²⁴⁰⁾
- PageBreaksBeforeItems ⁽²⁴⁴⁾
- RightMargin ⁽²⁴⁴⁾
- RTFOptions ⁽²⁴⁵⁾
- RTFReadProperties ⁽²⁴⁶⁾
- ▶ RVData ⁽²⁴⁷⁾
- RVFOptions ⁽²⁴⁷⁾
- RVFParaStylesReadMode ⁽²⁵¹⁾
- RVFTextStylesReadMode ⁽²⁵¹⁾
- ▶ RVFWarnings ⁽²⁵¹⁾
- SingleClick ⁽²⁵³⁾ (deprecated)
- SelectionHandlesVisible ⁽²⁵³⁾ (D2010+)
- **Style** ⁽²⁵³⁾
- StyleTemplateInsertMode ⁽²⁵⁴⁾
- TabNavigation ⁽²⁵⁵⁾
- TopMargin ⁽²⁵⁵⁾
- UseFMXThemes ⁽²⁵⁷⁾ [FMX]
- UseStyleTemplates ⁽²⁵⁶⁾
- UseVCLThemes ⁽²⁵⁷⁾ [VCL]
- VAlign ⁽²⁵⁷⁾
- VSmallStep ⁽²⁵⁷⁾ [VLC, LCL]
- WordWrap ⁽²⁵⁸⁾
- ZoomPercent ⁽²⁵⁸⁾ [FMX]

Derived from TRVScroller ⁽⁸¹³⁾

- BorderStyle ⁽⁸¹⁶⁾ [VCL]
- ▶ HScrollMax ⁽⁸¹⁶⁾
- HScrollPos ⁽⁸¹⁶⁾
- HScrollVisible ⁽⁸¹⁶⁾ [VCL,LCL]
- ▶ InplaceEditor ⁽⁸¹⁷⁾

- NoVScroll⁸¹⁷
- Tracking⁸¹⁸
- UseXPThemes⁸¹⁸ [VCL,LCL]
- ▶ VScrollMax⁸¹⁸
- VScrollPos⁸¹⁹
- VScrollVisible⁸¹⁹ [VCL,LCL]
- WheelStep⁸¹⁹ [VCL,LCL]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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
- ShowScrollBar [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
<i>rvddRVF</i>	'RichView Format'	RichView Format ⁽¹²⁴⁾
<i>rvddRTF</i>	'Rich Text Format'	RTF
<i>rvddText</i>	CF_TEXT	Plain ANSI text
<i>rvddUnicodeText</i>	CF_UNICODETEXT	Plain Unicode text
<i>rvddBitmap</i>	CF_DIB or CF_BITMAP	Bitmap
<i>rvddMetafile</i>	CF_ENHMETAFILE	Metafile
<i>rvddURL</i>	'UniformResourceLocator' or 'UniformResourceLocator W'	URL. If available, URL caption is read from CFSTR_FILEDESCRIPTOR format (otherwise caption = target). OnReadHyperlink ⁽³⁸⁸⁾ event occurs with DocFormat= <i>rvIfURL</i> . If this event is not processed, URL is inserted as a plain text.
<i>rvddFiles</i>	CF_HDROP	By default, RichViewEdit inserts: <ul style="list-style-type: none"> ▪ graphic files, ▪ *.RTF (only drag&drop), ▪ *.RVF (only drag&drop), ▪ *.TXT (only drag&drop). You can modify the default processing in OnDropFiles ⁽⁵⁷⁶⁾ event (only drag&drop).

Value	D&D Format	Meaning
<i>rvddHTML</i>	'HTML Format'	HTML

Example (processing `OnReadHyperlink` ⁽³⁸⁸⁾ with `rvddURL` support)

```
// OnReadHyperlink
procedure TMyForm.MyRichViewEditReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString(1032); DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag(1029);
  var ItemName: TRVUnicodeString(1032));
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(712);
begin
  Result := rve.CurTextStyleNo(474);
  TextStyle := TFontInfo.Create(nil);
  try
    TextStyle.Assign(711)(rve.Style(253).TextStyles(654)[Result]);
    TextStyle.Jump(714) := True;
    TextStyle.Color(701) := clBlue;
    TextStyle.Style(708) := TextStyle.Style+[fsUnderline];
    Result := rve.Style.FindTextStyle(661)(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⁽⁵³⁴⁾;
- CanPaste⁽⁴⁹⁸⁾;
- Drag&Drop⁽¹¹⁸⁾.

6.1.2.1.2 TRichViewEdit.ActualCurTextStyleNo

Index of the current text style in the collection of text styles. Can return `rvsDefStyle`⁽¹⁰⁵⁹⁾.

property `ActualCurTextStyleNo`: `Integer`;

(introduced in v1.8)

This is an index in the collection `Style`⁽²⁵³⁾.`TextStyles`⁽⁶⁵⁴⁾.

This property is the same as `CurTextStyleNo`⁽⁴⁷⁴⁾, but:

- read-only;
- does not process `rvsDefStyle`⁽¹⁰⁵⁹⁾, 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⁽¹³²⁾:

- using `OnSpellingCheck`⁽⁴⁰⁹⁾ and `OnGetSpellingSuggestions`⁽³⁷⁵⁾ 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`⁽⁴⁷⁹⁾. Assigning `False` to this property stops live spelling checking immediately.

Default value:

`False`

See also:

- Live spelling check in `TRichView`⁽¹³²⁾.

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`⁽²³⁷⁾-1.

If the caret is inside table cell, this property returns index of the table in the main document. Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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`⁽⁴⁸¹⁾.`CurItemNo`.

This property must be accessed only when the document is formatted.

When called for an empty editor (i.e. `ItemCount`⁽²³⁷⁾=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`⁽⁹⁸⁴⁾ unit instead.

See also properties:

- `OffsetInCurItem`⁽⁴⁷⁹⁾;
- `TopLevelEditor`⁽⁴⁸¹⁾;
- `CurItemStyle`⁽⁴⁷³⁾;

- `CurTextStyleNo` ⁽⁴⁷⁴⁾;
- `CurParaStyleNo` ⁽⁴⁷³⁾.

See also events:

- `OnCaretMove` ⁽⁵⁷²⁾.

See also:

- Inserting at the position of the caret ⁽¹¹⁴⁾;
- How to: move the caret to the beginning or to the end of document in `TRichViewEdit` ⁽¹⁰⁶⁵⁾.

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` ⁽⁴⁸¹⁾.`GetItemStyle` ⁽³⁰²⁾ (`TopLevelEditor` ⁽⁴⁸¹⁾.`CurItemNo` ⁽⁴⁷²⁾).

If the current item is a text item, this function returns zero or positive value: index in the collection of text styles (`Style` ⁽²⁵³⁾.`TextStyles` ⁽⁶⁵⁴⁾).

Otherwise, this function returns a type of this item: see `rsvXXX` constants ⁽¹⁰⁵⁹⁾.

This property must be accessed only when the document is formatted.

See also methods:

- `GetItemStyle` ⁽³⁰²⁾.

See also properties:

- `CurItemNo` ⁽⁴⁷²⁾;
- `CurTextStyleNo` ⁽⁴⁷⁴⁾.

See also:

- Item types ⁽⁸⁵⁾.

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` ⁽²⁵³⁾.`ParaStyles` ⁽⁶⁴⁸⁾.

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` ⁽⁵⁷³⁾ event occurs.

Usually you do not need not to assign value to this property directly, use `ApplyParaStyle` ⁽⁴⁹⁰⁾ or `ApplyParaStyleConversion` ⁽⁴⁹¹⁾ methods instead.

This property must be accessed only when the document is formatted.

See also methods:

- `ApplyParaStyle` ⁽⁴⁹⁰⁾;
- `ApplyParaStyleConversion` ⁽⁴⁹¹⁾.

See also properties:

- `CurTextStyleNo`⁽⁴⁷⁴⁾.

See also events:

- `OnCurParaStyleChanged`⁽⁵⁷³⁾.

See also:

- `Working with paragraphs`⁽⁹⁵⁾.

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`⁽²⁵³⁾.`TextStyles`⁽⁶⁵⁴⁾.

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`⁽⁵³⁸⁾ or `InsertText`⁽⁵³²⁾).

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`⁽⁵⁷⁴⁾ event occurs.

Value of this property is always positive. To get style (i.e. type) of item at the caret position, use `CurItemStyle`⁽⁴⁷³⁾ property.

The editor never sets this property to index of text style having `rvprDoNotAutoSwitch` in `Protection`⁽⁷⁰⁵⁾. You can do it yourself to insert a special symbol (by keyboard input, or `InsertText`⁽⁵³²⁾), but after that the component automatically switches this property back to the last used style without `rvprDoNotAutoSwitch` in `Protection`⁽⁷⁰⁵⁾.

Usually you do not need to assign value to this property directly, use `ApplyTextStyle`⁽⁴⁹⁴⁾ or `ApplyStyleConversion`⁽⁴⁹²⁾ instead.

This property must be accessed only when the document is formatted.

This property never returns `rvsDefStyle`⁽¹⁰⁵⁹⁾ 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`⁽¹⁰⁵⁹⁾ style, use `ActualCurTextStyleNo`⁽⁴⁷²⁾ property.

See also methods:

- `ApplyTextStyle`⁽⁴⁹⁴⁾;
- `ApplyStyleConversion`⁽⁴⁹²⁾.

See also properties:

- `CurItemStyle`⁽⁴⁷³⁾;
- `CurParaStyleNo`⁽⁴⁷³⁾;
- `ActualCurTextStyleNo`⁽⁴⁷²⁾.

See also events:

- `OnCurTextStyleChanged`⁽⁵⁷⁴⁾.

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`⁽²⁸⁸⁾ method (and is deactivated when you call `Clear`⁽²⁷⁹⁾). It's not recommended to use the custom caret feature together with image animation⁽¹¹⁹⁾.

Default value:

0 (the system caret is used)

See also events:

- `OnMeasureCustomCaret`⁽⁵⁷⁸⁾;
- `OnDrawCustomCaret`⁽⁵⁷⁵⁾.

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`⁽¹⁰³³⁾;

(introduced in version 14)

The following methods use this property:

- `PasteBitmap`⁽⁵³⁵⁾
- `PasteMetafile`⁽⁵³⁷⁾
- `PasteGraphicFile`⁽⁵³⁵⁾
- `Paste`⁽⁵³⁴⁾ (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
<code>rvoDoNotWantReturns</code>	If set, editor ignores Enter keys.

Option	Meaning
<i>rvoDoNotWantShiftReturns</i>	If set, Shift + Enter does nothing (by default, it inserts a line break inside paragraph – noticeable in paragraphs with bullets or borders)
<i>rvoWantTabs</i>	If set (default), editor accepts Tab key (Ctrl + Tab always works). Tabs can be inserted as several spaces, or as a tabulator ⁽¹⁶²⁾ (special item type), depending on the value of <i>SpacesInTab</i> ⁽⁶⁵²⁾ property of the linked TRVStyle component.
<i>rvoNoImageResize</i>	Prevents resizing pictures ⁽¹⁶³⁾ and controls ⁽¹⁶⁸⁾ with mouse.
<i>rvoAlt0CodesUseKeyboardCodepage</i>	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
<i>rvoNoCaretHighlightJumps</i>	If set, hyperlinks will not be highlighted (with <i>HoverColor</i> ⁽⁷⁰²⁾ and <i>HoverBackColor</i> ⁽⁷⁰²⁾) when the caret is moved inside them.
<i>rvoHideReadOnlyCaret</i>	If set, the editor hides the caret in read-only mode ⁽⁴⁸⁰⁾

Drag&drop options

Option	Meaning
<i>rvoDragDropPicturesFromLinks</i>	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.

Option	Meaning
	<p>The most probably, the link points to a remote image, so it needs to be downloaded before insertion. You can do it in <code>OnImportPicture</code> ⁽³⁷⁸⁾ event.</p> <p>If <code>rvoAssignImageFileNames</code> in <code>Options</code> ⁽²⁴⁰⁾, the component assigns a "file name" property for this image.</p>

Hypertext options

Option	Meaning
<code>rvoCtrlJumps</code>	<p>If set (default), the editor switches to the hypertext mode (highlighting hypertext items under mouse cursor, using hypertext cursors, generating <code>OnJump</code> ⁽³⁸²⁾ and <code>OnRVMouseMove</code> ⁽³⁹⁵⁾) when the user presses and holds Ctrl * key</p> <p>The editor switches back to the editing mode when the user releases Ctrl * key or changes document, or if the editor loses focus.</p> <p>* on macOS, Command is used instead of Ctrl.</p>
<code>rvoNoReadOnlyJumps</code>	<p>If not set (default), the editor automatically switches to the hypertext mode when it becomes read-only.</p> <p>If set, hyperlinks in a read-only mode work like in a normal mode.</p>

Protection options

Option	Meaning
<code>rvoFastDeleteProtectedText</code>	<p>Specifies how a selected fragment that contains a modify-protected* text partially is deleted.</p> <p>If not set (default), deletion is not allowed (the user must select the protected text completely to delete it).</p> <p>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).</p>

* the protected text's style has `rvprModifyProtect` in `Protection` ⁽⁷⁰⁵⁾, and does not have `rvprDeleteProtect` option there.

Other options

Option	Meaning
<code>rvoClearTagOnStyleApp</code>	If set, editor clears <i>tags</i> ⁽⁹¹⁾ (set to "") of all text items when changing their style index using <code>ApplyTextStyle</code> ⁽⁴⁹⁴⁾ or <code>ApplyStyleConversion</code> ⁽⁴⁹²⁾ method.
<code>rvoAutoSwitchLang</code>	Obsolete, does nothing.

Default value (changed in v1.9):

[`rvoWantTabs`, `rvoCtrlJumps`]

See also:

- Options⁽²⁴⁰⁾.

See also events:

- OnJump⁽³⁸²⁾;
- OnRVMouseMove⁽³⁹⁵⁾.

See also:

- TRVStyle.SpacesInTab⁽⁶⁵²⁾;
- RichView Hypertext⁽⁹²⁾;
- Tags⁽⁹¹⁾.

6.1.2.1.11 TRichViewEdit.ForceFieldHighlight

Allows highlighting label items⁽¹⁷⁶⁾ (including numbered sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, sidenotes⁽¹⁸¹⁾) even when the editor is not focused.

property `ForceFieldHighlight`: Boolean;

(Introduced in version 15)

The editor can highlight label items with `Style`⁽²⁵³⁾.`FieldHighlightColor`⁽⁶³⁷⁾.

There are three modes possible, depending on `Style`⁽²⁵³⁾.`FieldHighlightType`⁽⁶³⁸⁾: 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
<i>rvlspManualStart</i>	Live spelling check can be started only by <code>StartLiveSpelling</code> ³⁶⁷ .
<i>rvlspOnChange</i>	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 TRichView¹³².

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`²⁷⁹ resets it to *False*.

Any editing operation, including calling any editor-style method (which invokes `OnChange`⁵⁷² 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`⁴⁷² 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 `OffsetInCurlItem=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`⁽³⁶⁵⁾.

If the caret is inside table cell, this property returns 1 (position after the table). Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾ and `TopLevelEditor`⁽⁴⁸¹⁾.`OffsetInCurlItem`.

This property must be accessed only when the document is formatted.

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾;
- `TopLevelEditor`⁽⁴⁸¹⁾.

See also events:

- `OnCaretMove`⁽⁵⁷²⁾.

See also methods:

- `SetSelectionBounds`⁽³⁶⁵⁾;
- `GetSelectionBounds`⁽³¹²⁾.

See also:

- Inserting at the position of the caret⁽¹¹⁴⁾;
- How to: move the caret to the beginning or to the end of document in `TRichViewEdit`⁽¹⁰⁶⁵⁾.

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`⁽⁴⁷⁵⁾.

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`⁽⁵⁷²⁾.

6.1.2.1.17 TRichViewEdit.SmartPopupProperties

Properties for "smart popups" (analog of Microsoft Office's "smart tags" buttons).

property `SmartPopupProperties`: `TRVSmartPopupProperties`⁽⁵⁸⁸⁾;

(introduced in version 10)

See `TRVSmartPopupProperties`⁽⁵⁸⁸⁾ for details.

See also properties:

- SmartPopupVisible⁽⁴⁸¹⁾.

See also events:

- OnSmartPopupClick⁽⁵⁸⁵⁾.

6.1.2.1.18 TRichViewEdit.SmartPopupVisible

Shows/hides "smart popup"⁽⁵⁸⁸⁾ 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⁽⁵⁷²⁾ 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⁽⁴⁸⁰⁾.

See also events:

- OnSmartPopupClick⁽⁵⁸⁵⁾.

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⁽⁸¹⁷⁾ 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

Value	Meaning
-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 ⁽¹¹⁵⁾.

6.1.2.2 Methods

Derived from TCustomRichViewEdit

AddSpellingMenuItems ⁽⁴⁸⁸⁾ [FMX]
 AdjustControlPlacement ⁽⁴⁸⁹⁾
 AdjustControlPlacement2 ⁽⁴⁸⁹⁾
 ApplyListStyle ⁽⁴⁹⁰⁾
 ApplyParaStyle ⁽⁴⁹⁰⁾
 ApplyParaStyleConversion ⁽⁴⁹¹⁾
 ApplyParaStyleTemplate ⁽⁴⁹¹⁾
 ApplyStyleConversion ⁽⁴⁹²⁾
 ApplyStyleTemplate ⁽⁴⁹³⁾
 ApplyTextStyle ⁽⁴⁹⁴⁾
 ApplyTextStyleTemplate ⁽⁴⁹⁴⁾
 BeginItemModify ⁽⁴⁹⁵⁾
 BeginUndoCustomGroup ⁽⁴⁹⁶⁾
 BeginUndoCustomGroup2 ⁽⁴⁹⁶⁾
 BeginUndoGroup ⁽⁴⁹⁷⁾
 BeginUndoGroup2 ⁽⁴⁹⁷⁾
 BeginUpdate ⁽⁴⁹⁸⁾
 CanChange ⁽⁴⁹⁸⁾
 CanPaste ⁽⁴⁹⁸⁾
 CanPasteHTML ⁽⁴⁹⁹⁾
 CanPasteRTF ⁽⁴⁹⁹⁾
 CanPasteRVF ⁽⁴⁹⁹⁾
 Change ⁽⁴⁹⁹⁾

-  `ChangeListLevels` ⁽⁵⁰⁰⁾
-  `ChangeStyleTemplates` ⁽⁵⁰⁰⁾
- `Clear` ⁽⁵⁰¹⁾
-  `ClearTextFlow` ⁽⁵⁰¹⁾
- `ClearUndo` ⁽⁵⁰¹⁾
-  `ConvertToHotPicture` ⁽⁵⁰¹⁾
-  `ConvertToPicture` ⁽⁵⁰²⁾
- `Create` ⁽⁵⁰²⁾
-  `CutDef` ⁽⁵⁰²⁾
-  `DeleteSelection` ⁽⁵⁰³⁾
- `Destroy` ⁽⁵⁰³⁾
- `EndItemModify` ⁽⁵⁰³⁾
- `EndUndoGroup2` ⁽⁴⁹⁷⁾
- `EndUndoCustomGroup2` ⁽⁴⁹⁶⁾
- `EndUpdate` ⁽⁵⁰³⁾
- `GetCurrentCheckpointAtCaret` ⁽⁵⁰³⁾
- `GetCurrentBreakInfo` ⁽⁵⁰⁴⁾
- `GetCurrentBulletInfo` ⁽⁵⁰⁵⁾
- `GetCurrentCheckpoint` ⁽⁵⁰⁶⁾
- `GetCurrentControlInfo` ⁽⁵⁰⁶⁾
- `GetCurrentHotspotInfo` ⁽⁵⁰⁷⁾
- `GetCurrentItemExtraIntProperty` ⁽⁵¹⁰⁾
- `GetCurrentItemExtraIntPropertyEx` ⁽⁵¹⁰⁾
- `GetCurrentItemExtraStrProperty` ⁽⁵¹⁰⁾
- `GetCurrentItemExtraStrPropertyEx` ⁽⁵¹⁰⁾
- `GetCurrentItem` ⁽⁵⁰⁸⁾
- `GetCurrentItemEx` ⁽⁵⁰⁹⁾
- `GetCurrentItemText -A -W` ⁽⁵¹¹⁾
- `GetCurrentItemVAlign` ⁽⁵¹²⁾
- `GetCurrentLineCol` ⁽⁵¹²⁾
- `GetCurrentMisspelling` ⁽⁵¹³⁾
- `GetCurrentPictureInfo` ⁽⁵¹³⁾
- `GetCurrentTag` ⁽⁵¹⁴⁾
- `GetCurrentTextInfo` ⁽⁵¹⁴⁾
-  `InsertBreak` ⁽⁵¹⁵⁾
-  `InsertBullet` ⁽⁵¹⁵⁾
-  `InsertCheckpoint` ⁽⁵¹⁶⁾
-  `InsertControl` ⁽⁵¹⁷⁾
-  `InsertDocXFromFileEd` ⁽⁵¹⁸⁾ (D2009+ or with Delphi ZLib)
-  `InsertDocXFromStreamEd` ⁽⁵¹⁸⁾ (D2009+ or with Delphi ZLib)
-  `InsertHotPicture` ⁽⁵¹⁹⁾
-  `InsertHotspot` ⁽⁵²⁰⁾
-  `InsertHTMLFromFileEd` ⁽⁵²¹⁾
-  `InsertHTMLFromStreamEd` ⁽⁵²¹⁾
-  `InsertItem` ⁽⁵²²⁾
-  `InsertMarkdownFromFileEd` ⁽⁵²³⁾
-  `InsertMarkdownFromStreamEd` ⁽⁵²⁴⁾

- I InsertOEMTextFromFile ⁵²⁵
- I InsertPageBreak ⁵²⁵
- I InsertPicture ⁵²⁶
- I InsertRTFFromFileEd ⁵²⁷
- I InsertRTFFromStreamEd ⁵²⁸
- I InsertRVFFromFileEd ⁵²⁸
- I InsertRVFFromStreamEd ⁵²⁹
- I InsertStringTag -A -W ⁵³⁰
- I InsertTab ⁵³¹
- I InsertText -A -W ⁵³²
- I InsertTextFromFile -W ⁵³³
- MoveCaret ⁵³⁴
- I Paste ⁵³⁴
- I PasteBitmap ⁵³⁵
- I PasteHTML ⁵³⁶
- I PasteGraphicFile ⁵³⁵
- I PasteMetafile ⁵³⁷
- I PasteRTF ⁵³⁷
- I PasteRVF ⁵³⁸
- I PasteText -A -W ⁵³⁸
- I PasteURL ⁵³⁹
- I Redo ⁵⁴⁰
- RedoAction ⁵⁴⁰
- RedoName ⁵⁴⁰
- I RemoveCheckpointAtCaret ⁵⁴¹
- I RemoveCheckpointEd ⁵⁴¹
- I RemoveCurrentCheckpoint ⁵⁴¹
- I RemoveCurrentPageBreak ⁵⁴²
- I RemoveLists ⁵⁴²
- I ResizeControl ⁵⁴²
- I ResizeCurrentControl ⁵⁴³
- SearchText -A -W ⁵⁴³
- SelectCurrentLine ⁵⁴⁵
- SelectCurrentWord ⁵⁴⁵
- I SetBackgroundImageEd ⁵⁴⁶
- I SetBreakInfoEd ⁵⁴⁷
- I SetBulletInfoEd ⁵⁴⁸
- I SetCheckpointInfoEd ⁵⁴⁹
- I SetControlInfoEd ⁵⁵⁰
- I SetCurrentBreakInfo ⁵⁵¹
- I SetCurrentBulletInfo ⁵⁵²
- I SetCurrentCheckpointInfo ⁵⁵³
- I SetCurrentControlInfo ⁵⁵⁴
- I SetCurrentHotspotInfo ⁵⁵⁵
- I SetCurrentItemExtraIntProperty ⁵⁵⁶

-  SetCurrentItemExtraIntPropertyEx ⁽⁵⁵⁶⁾
-  SetCurrentItemExtraStrProperty ⁽⁵⁵⁷⁾
-  SetCurrentItemExtraStrPropertyEx ⁽⁵⁵⁷⁾
-  SetCurrentItemText -A -W ⁽⁵⁵⁷⁾
-  SetCurrentItemVAlign ⁽⁵⁵⁸⁾
-  SetCurrentPictureInfo ⁽⁵⁵⁹⁾
-  SetCurrentTag ⁽⁵⁶⁰⁾
-  SetFloatPropertyEd ⁽⁵⁴⁵⁾
-  SetHotspotInfoEd ⁽⁵⁶⁰⁾
-  SetIntPropertyEd ⁽⁵⁴⁵⁾
-  SetItemExtraIntPropertyEd ⁽⁵⁶²⁾
-  SetItemExtraIntPropertyExEd ⁽⁵⁶²⁾
-  SetItemExtraStrPropertyEd ⁽⁵⁶²⁾
-  SetItemExtraStrPropertyExEd ⁽⁵⁶²⁾
-  SetItemTagEd ⁽⁵⁶³⁾
-  SetItemTextEd -A -W ⁽⁵⁶⁴⁾
-  SetItemVAlignEd ⁽⁵⁶⁵⁾
-  SetPictureInfoEd ⁽⁵⁶⁵⁾
-  SetStrPropertyEd ⁽⁵⁴⁵⁾
- SetUndoGroupMode ⁽⁵⁶⁷⁾
-  Undo ⁽⁵⁶⁷⁾
- UndoAction ⁽⁵⁶⁷⁾
- UndoName ⁽⁵⁶⁸⁾

Derived from TCustomRichView ⁽²¹⁰⁾

-  AddBreak ⁽²⁶²⁾
-  AddBullet ⁽²⁶³⁾
-  AddCheckpoint ⁽²⁶⁴⁾
-  AddControl ⁽²⁶⁵⁾
-  AddFmt ⁽²⁶⁶⁾
-  AddHotPicture ⁽²⁶⁷⁾
-  AddHotspot ⁽²⁶⁸⁾
-  AddItem ⁽²⁶⁹⁾
-  AddNL -A -W ⁽²⁷⁰⁾
-  AddPicture ⁽²⁷²⁾
-  AddTab ⁽²⁷³⁾
-  AddTextNL -A -W ⁽²⁷⁴⁾
-  AppendFrom ⁽²⁷⁵⁾
-  AppendRVFFromStream ⁽²⁷⁶⁾
- AssignSoftPageBreaks ⁽²⁷⁷⁾
- BeginOleDrag ⁽²⁷⁸⁾
-  Clear ⁽²⁷⁹⁾
- ClearLiveSpellingResults ⁽²⁷⁹⁾
- ClearSoftPageBreaks ⁽²⁷⁹⁾
- ClientToDocument, DocumentToClient ⁽²⁸⁰⁾
- ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits ⁽²⁸⁰⁾

Copy²⁸¹
CopyDef²⁸¹
CopyImage²⁸¹
CopyRTF²⁸²
CopyRVF²⁸²
CopyText -A -W²⁸³
Create²⁸³
 DeleteItems²⁸³
DeleteMarkedStyles²⁸⁴
DeleteParas²⁸⁵
 DeleteSection²⁸⁵
DeleteUnusedStyles²⁸⁶
Deselect²⁸⁶
Destroy²⁸⁶
FindCheckpointByName²⁸⁷
FindCheckpointByTag²⁸⁷
FindControlItemNo²⁸⁷
Format²⁸⁸
FormatTail²⁸⁸
GetBreakInfo²⁸⁹
GetBulletInfo²⁸⁹
GetCheckpointByNo²⁹¹
GetCheckpointInfo²⁹¹
GetCheckpointItemNo²⁹¹
GetCheckpointNo²⁹²
GetCheckpointXY²⁹²
GetCheckpointY²⁹²
GetCheckpointYEx²⁹³
GetControlInfo²⁹³
GetFirstCheckpoint²⁹⁴
GetFocusedItem²⁹⁵
GetHotspotInfo²⁹⁵
GetItem²⁹⁶
GetItemAt²⁹⁷
GetItemCheckpoint²⁹⁸
GetItemCoords²⁹⁸
GetItemCoordsEx²⁹⁹
GetItemExtraIntProperty³⁰⁰
GetItemExtraIntPropertyEx³⁰⁰
GetItemExtraStrProperty³⁰⁰
GetItemExtraStrPropertyEx³⁰⁰
GetItemNo³⁰¹
GetItemPara³⁰¹
GetItemStyle³⁰²
GetItemTag³⁰²
GetItemText -A -W³⁰³
GetItemVAlign³⁰³

GetJumpPointLocation³⁰⁴
GetJumpPointY³⁰⁵
GetLastCheckpoint³⁰⁵
GetLineNo³⁰⁶
GetListMarkerInfo³⁰⁷
GetNextCheckpoint³⁰⁷
GetOffsAfterItem³⁰⁸
GetOffsBeforeItem³⁰⁹
GetPictureInfo³¹⁰
GetPrevCheckpoint³¹¹
GetRealDocumentPixelsPerInch³¹¹
GetSelectedImage³¹²
GetSelectionBounds³¹²
GetSelText -W³¹³
GetTextInfo³¹³
GetWordAt -A -W³¹⁴
 InsertRVFFromStream³¹⁶
IsFromNewLine³¹⁷
IsParaStart³¹⁸
LiveSpellingValidateWord³¹⁸
 LoadDocX³¹⁸ (D2009+ or with Delphi ZLib)
 LoadDocXFromStream³¹⁹ (D2009+ or with Delphi ZLib)
 LoadFromFile³²⁰
 LoadFromFileEx³²⁰
 LoadFromStream³²¹
 LoadFromStreamEx³²¹
 LoadHTML³²²
 LoadHTMLFromStream³²⁴
 LoadMarkdown³²⁵
 LoadMarkdownFromStream³²⁵
 LoadRTF³²⁶
 LoadRTFFromStream³²⁷
 LoadRVF³²⁸
 LoadRVFFromStream³²⁹
 LoadText -W³³⁰
 LoadTextFromStream -W³³¹
MarkStylesInUse³³¹
RefreshListMarkers³³²
Reformat³³²
 RemoveCheckpoint³³²
ResetAnimation³³³
SaveDocX³³³
SaveDocXToStream³³⁴
SaveHTML³³⁵
SaveHTMLToStream³³⁸
SavePicture³⁴²
SaveRTF³⁴³

SaveRTFToStream ⁽³⁴⁴⁾
 SaveRVF ⁽³⁴⁴⁾
 SaveRVFToStream ⁽³⁴⁴⁾
 SaveText -W ⁽³⁴⁵⁾
 SaveTextToStream -W ⁽³⁴⁵⁾
 SearchText -A -W ⁽³⁴⁶⁾
 SelectAll ⁽³⁴⁸⁾
 SelectControl ⁽³⁴⁸⁾
 SelectionExists ⁽³⁴⁹⁾
 SelectWordAt ⁽³⁴⁹⁾
 SetAddParagraphMode ⁽³⁵⁰⁾
 SetBreakInfo ⁽³⁵¹⁾
 SetBulletInfo ⁽³⁵²⁾
 SetCheckpointInfo ⁽³⁵³⁾
 SetControlInfo ⁽³⁵⁴⁾
 SetFooter ⁽³⁵⁵⁾
 SetHeader ⁽³⁵⁶⁾
 SetHotspotInfo ⁽³⁵⁷⁾
 SetItemExtraIntProperty ⁽³⁵⁸⁾
 SetItemExtraIntPropertyEx ⁽³⁵⁸⁾
 SetItemExtraStrProperty ⁽³⁵⁹⁾
 SetItemExtraStrPropertyEx ⁽³⁵⁹⁾
 SetItemTag ⁽³⁶⁰⁾
 SetItemText -A -W ⁽³⁶⁰⁾
 SetItemVAlign ⁽³⁶¹⁾
 SetListMarkerInfo ⁽³⁶²⁾
 SetPictureInfo ⁽³⁶³⁾
 SetSelectionBounds ⁽³⁶⁵⁾
 StartAnimation ⁽³⁶⁶⁾
 StartLiveSpelling ⁽³⁶⁷⁾
 StopAnimation ⁽³⁶⁷⁾
 StoreSearchResult ⁽³⁶⁷⁾
 UpdatePaletteInfo ⁽³⁶⁸⁾

Derived from TRVScroller ⁽⁸¹³⁾

ScrollTo ⁽⁸²⁰⁾

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`³⁷⁵ event;
- from a platform spelling check service, if it is available for the current platform.

6.1.2.2.2 TRichViewEdit.AdjustControlPlacement

Returns the control¹⁶⁸ 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`⁵⁴³ (or `ResizeControl`⁵⁴²) 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`⁴⁸⁹ instead of this method, because it can work with controls inserted in table¹⁹⁰ 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`⁵⁴³ (or `ResizeControl`⁵⁴²) 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`⁴⁸⁹.

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 (Style²⁵³.ListStyles⁶⁴⁶).

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[AListNo].Levels⁷³²).

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 SetListMarkerInfo³⁶²).

See also:

- RemoveLists⁵⁴²;
- ChangeListLevels⁵⁰⁰.

See also methods of TCustomRichView:

- SetListMarkerInfo³⁶²;
- GetListMarkerInfo³⁰⁷.

6.1.2.2.5 TRichViewEdit.ApplyParaStyle

Applies the specified paragraph style.

```
procedure ApplyParaStyle(ParaStyleNo: Integer);
```

ParaStyleNo – index in the collection of paragraph styles (Style²⁵³.ParaStyles⁶⁴⁸).

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 Options⁷²³, this method does not change their styles.

Method type:  editing-style.

If you want to implement commands like "change alignment", "increase indent" use ApplyParaStyleConversion⁴⁹¹ instead.

See also properties:

- CurParaStyleNo⁴⁷³.

See also methods:

- [ApplyTextStyle](#) ⁴⁹⁴;
- [ApplyParaStyleConversion](#) ⁴⁹¹.

See also:

- [TParaInfo.Options](#) ⁷²³;
- [Paragraphs overview](#) ⁹⁵.

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](#) ⁵⁸² event for each selected item. Using this method, you can implement commands like "change alignment" or "increase indent".

This method ignores any paragraph protection ⁷²³, it can change paragraph style even if *rvpaoStyleProtect* is included in [Options](#) ⁷²³ of existing paragraphs' styles. If you want to respect paragraph style protection, check this option yourself in [OnParaStyleConversion](#) ⁵⁸² event.

If **Recursive** = *False*, [OnParaStyleConversion](#) ⁵⁸² is not called for cells of selected tables ¹⁷³ (except for the case of multicell selection in a single table).

Method type:  editing-style.

Demo:

- `Demos*\Editors\Editor 2\`

See also:

- [ApplyStyleConversion](#) ⁴⁹²;
- [ApplyParaStyle](#) ⁴⁹⁰.

See also events:

- [OnParaStyleConversion](#) ⁵⁸².

See also:

- [Paragraphs overview](#) ⁹⁵.

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 ⁶⁴⁸ of items in the selected paragraphs.

Parameters:

TemplateNo – index in the collection `Style(253).StyleTemplates(652)`, or the value -1. The value -1 means a format clearing (see below). The style template must have `Kind(735) = 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⁽¹⁷³⁾ (except for the case of multicell selection in a single table).

This method does nothing if `UseStyleTemplates(256)=False` and **TemplateNo**>=0.

If paragraphs are formatted with styles having `rvpaoStyleProtect` in their `Options(723)`, 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(696) = -1`.

If `UseStyleTemplates(256)=False` and **TemplateNo**<0, the results are identical to calling `ApplyParaStyle(490)(0)`.

Modified properties

Properties of a paragraph style⁽⁷²⁷⁾ are considered modified, if they have values different from values of the corresponding properties of the current style template (referred in `StyleTemplateId(696)` property). If a style template is not assigned, the properties are considered modified if they have non-default values. The method ignores `ModifiedProperties(729)` property (it calculates this property itself).

Method type:  editing-style.

See also:

- `ApplyTextStyleTemplate(494)`;
- `ApplyParaStyleConversion(491)`;
- `ApplyParaStyle(490)`.

6.1.2.2.8 TRichViewEdit.ApplyStyleConversion

Applies a custom conversion procedure to styles of the selected text items⁽¹⁶¹⁾, tabs⁽¹⁶²⁾, label items⁽¹⁷⁶⁾ (and item of all types inherited from label items).

```
procedure ApplyStyleConversion(UserData: Integer;  
Recursive: Boolean = True);
```

This procedure calls `OnStyleConversion(585)` event for each selected text item and for the current text style⁽⁴⁷⁴⁾. Using this method, you can implement commands like "make bold" or "change font name".

This method ignores any text protection⁽⁷⁰⁵⁾, it can change text style even if *rvprStyleProtect* is included in Protection⁽⁷⁰⁵⁾ of existing text items' styles. If you want to respect text style protection, check this protection option yourself in OnStyleConversion⁽⁵⁸⁵⁾ event.

If **Recursive** = *False*, OnStyleConversion⁽⁵⁸⁵⁾ is not called for cells of selected tables⁽¹⁷³⁾ (except for the case of multicell selection in a single table).

Method type:  editing-style.

Demo:

- Demos*\Editors\Editor 2\

See also:

- ApplyParaStyleConversion⁽⁴⁹¹⁾;
- ApplyTextStyle⁽⁴⁹⁴⁾.

See also events:

- OnStyleConversion⁽⁵⁸⁵⁾.

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⁽⁶⁵⁴⁾ or paragraph styles⁽⁶⁴⁸⁾ of items in the selected paragraphs. The result depends on the style template's Kind⁽⁷³⁵⁾ and the selection.

Parameters:

TemplateNo – index in the collection Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾, 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⁽¹⁷³⁾ (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates⁽²⁵⁶⁾=*False* and **TemplateNo**>=0.

The method works according to the following rules:

- if Kind⁽⁷³⁵⁾ of the specified style template = *rvstkText*, it is applied to the selected text;
- if Kind⁽⁷³⁵⁾ of the specified style template = *rvstkPara*, it is applied to the selected paragraphs; the only exception is "Normal" style template⁽⁶⁸⁹⁾;
- if **TemplateNo**=-1, or Kind⁽⁷³⁵⁾ 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⁽⁴⁹⁴⁾ or ApplyParaStyleTemplate⁽⁴⁹¹⁾ called with **ResetAdditionalFormatting**=*True*.

The method skips paragraph styles having *rvpaoStyleProtect* in their Options⁽⁷²³⁾, and text styles having *rvprStyleProtect* in their Protection⁽⁷⁰⁵⁾.

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`⁽⁴⁹⁴⁾ and `ApplyParaStyleTemplate`⁽⁴⁹¹⁾.

Method type:  editing-style.

See also:

- `ChangeStyleTemplates`⁽⁵⁰⁰⁾.

6.1.2.2.10 TRichViewEdit.ApplyTextStyle

Applies the specified text style to the selected items (text items⁽¹⁶¹⁾, tabs⁽¹⁶²⁾, label items⁽¹⁷⁶⁾)

procedure `ApplyTextStyle(TextStyleNo: Integer);`

TextStyleNo – index in the collection of text styles (`Style`⁽²⁵³⁾.`TextStyles`⁽⁶⁵⁴⁾).

If selection is empty, this method only sets `CurTextStyleNo`⁽⁴⁷⁴⁾ to **TextStyleNo**. If the selection is not empty, the method changes style of text items⁽¹⁶¹⁾ in the selection.

If the existing text items are formatted with styles having `rvprStyleProtect` in their `Protection`⁽⁷⁰⁵⁾, 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`⁽⁴⁹²⁾ instead of this method.

See also properties:

- `CurTextStyleNo`⁽⁴⁷⁴⁾;
- `EditorOptions`⁽⁴⁷⁵⁾.

See also methods:

- `ApplyParaStyle`⁽⁴⁹⁰⁾.

6.1.2.2.11 TRichViewEdit.ApplyTextStyleTemplate

Applies the specified style template to the selected text items⁽¹⁶¹⁾, tabs⁽¹⁶²⁾, label items⁽¹⁷⁶⁾ (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⁽⁶⁵⁴⁾ of the selected items.

Parameters:

TemplateNo – index in the collection `Style`⁽²⁵³⁾.`StyleTemplates`⁽⁶⁵²⁾, or the value -1. The value -1 means a format clearing (see below). The style template must have `Kind`⁽⁷³⁵⁾ = `rvstkParaText` or `rvstkText`. The only exception is "Normal" style template⁽⁶⁸⁹⁾: 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⁽¹⁷³⁾ (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates⁽²⁵⁶⁾=*False* and **TemplateNo**>=0.

If items are formatted with text styles having *rvprStyleProtect* in their Protection⁽⁷⁰⁵⁾, this method skips them.

Format clearing

A format clearing means resetting all text attributes to default values. If a paragraph style template (ParaStyleTemplateId⁽⁷⁰⁵⁾) is assigned, it defines these default values. A special processing is performed for hyperlinks (i.e. text styles with Jump⁽⁷¹⁴⁾=*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⁽²⁵⁶⁾=*False* and **TemplateNo**<0, the results are identical to calling ApplyTextStyle⁽⁴⁹⁴⁾(0).

Modified properties

Properties of a text style⁽⁷¹²⁾ are considered modified, if they have values different from values of the corresponding properties of the current style templates (referred in StyleTemplateId⁽⁶⁹⁶⁾ and ParaStyleTemplateId⁽⁷⁰⁵⁾ properties). If none of these style templates are assigned, the properties are considered modified if they have non-default values. The method ignores ModifiedProperties⁽⁷¹⁴⁾ property (it calculates this property itself).

Method type:  editing-style.

See also:

- ApplyParaStyleTemplate⁽⁴⁹¹⁾;
- ApplyStyleConversion⁽⁴⁹²⁾;
- ApplyTextStyle⁽⁴⁹⁴⁾.

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(998));
```

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⁽⁵⁰³⁾ after performing modifications.

Usually this function is used when modifying tables⁽⁸⁴⁶⁾.

This method must be called only when the document is formatted.

Example is in Table Operations Overview¹⁹⁹

6.1.2.2.13 TRichViewEdit.BeginUndoCustomGroup

Begins a new undo action.

```
procedure BeginUndoCustomGroup(const Name: TRVUnicodeString1032);
```

(changed in version 18)

Begins a new undo action.

Usually followed by SetUndoGroupMode⁵⁶⁷(True) ... SetUndoGroupMode(False), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as BeginUndoGroup⁴⁹⁷, 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 TopLevelEditor⁴⁸¹).

See also:

- Undo in RichViewEdit¹¹⁵.

6.1.2.2.14 TRichViewEdit.BeginUndoCustomGroup2, EndUndoCustomGroup2

An alternative method to group operation for undo.

```
procedure BeginUndoCustomGroup2(const Name: TRVUnicodeString1032);
```

```
procedure EndUndoCustomGroup2(const Name: TRVUnicodeString1032);
```

(introduced in version 18)

The recommended and most efficient way to group operations (to undo them in a single step) is:

```
Edit.TopLevelEditor481.BeginUndoCustomGroup496(OperationName);
```

```
Edit.TopLevelEditor.SetUndoGroupMode567(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 EndUndoGroup2⁴⁹⁷, 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 (UndoLimit⁴⁸¹ must be equal to -1), otherwise they do nothing.

Unlike the first set of method (which must be called for TopLevelEditor⁴⁸¹), 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`⁽¹¹⁵⁾

6.1.2.2.15 `TRichViewEdit.BeginUndoGroup`

Begins a new undo action.

```
procedure BeginUndoGroup(UndoType: TRVUndoType(1031));
```

Begins a new undo action.

Usually followed by `SetUndoGroupMode`⁽⁵⁶⁷⁾ (*True*) ... `SetUndoGroupMode`(*False*), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as `BeginUndoCustomGroup`⁽⁴⁹⁶⁾, 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`⁽⁴⁸¹⁾).

See also:

- Undo in `RichViewEdit`⁽¹¹⁵⁾, includes an example.

6.1.2.2.16 `TRichViewEdit.BeginUndoGroup2, EndUndoGroup2`

An alternative method to group operations for undo.

```
procedure BeginUndoGroup2(UndoType: TRVUndoType(1031));
```

```
procedure EndUndoGroup2(UndoType: TRVUndoType(1031));
```

(introduced in version 18)

The recommended and most efficient way to group operations (to undo them in a single step) is:

```
Edit.TopLevelEditor(481).BeginUndoGroup(497)(UndoType);
Edit.TopLevelEditor.SetUndoGroupMode(567)(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`⁽⁴⁹⁶⁾, 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⁽⁴⁸¹⁾ must be equal to -1), otherwise they do nothing.

Unlike the first set of methods (which must be called for TopLevelEditor⁽⁴⁸¹⁾), 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⁽¹¹⁵⁾, includes an example.

6.1.2.2.17 TRichViewEdit.BeginUpdate

Suspends repainting until EndUpdate⁽⁵⁰³⁾ 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⁽⁴⁸⁰⁾ property.

Usually this function is called before performing operations on tables⁽⁸⁴⁶⁾ to prevent modifying editor in read-only state.

Additionally, if live spelling checking⁽¹³²⁾ is active, this method pauses a live spelling checking thread (until the call of Change⁽⁴⁹⁹⁾), to prevent errors when accessing data that are being modified.

For TDBRichViewEdit⁽⁶⁰⁶⁾, 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⁽⁴⁷⁰⁾ property.

function CanPaste: Boolean;

See also methods:

- Paste⁽⁵³⁴⁾;
- CanPasteRVF⁽⁴⁹⁹⁾;
- CanPasteHTML⁽⁴⁹⁹⁾;
- CanPasteRTF⁽⁴⁹⁹⁾.

See also:

- Working with Clipboard⁽¹⁰⁸⁾.

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⁽⁵³⁶⁾;
- CanPaste⁽⁴⁹⁸⁾.

See also:

- Working with Clipboard⁽¹⁰⁸⁾

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⁽⁵³⁷⁾;
- CanPaste⁽⁴⁹⁸⁾.

See also:

- Working with Clipboard⁽¹⁰⁸⁾

6.1.2.2.22 TRichViewEdit.CanPasteRVF

Returns *True* if the Clipboard contains data in RVF (RichView Format⁽¹²⁴⁾).

```
function CanPasteRVF: Boolean;
```

See also methods:

- PasteRVF⁽⁵³⁸⁾;
- CanPaste⁽⁴⁹⁸⁾.

See also:

- Working with Clipboard⁽¹⁰⁸⁾;
- RVF⁽¹²⁴⁾.

6.1.2.2.23 TRichViewEdit.Change

Generates OnChange⁽⁵⁷²⁾ event.

```
procedure Change;
```

This method also assigns *True* to Modified⁽⁴⁷⁹⁾ 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`⁴⁹⁰;
- `RemoveLists`⁵⁴².

See also methods of TCustomRichView:

- `SetListMarkerInfo`³⁶²;
- `GetListMarkerInfo`³⁰⁷.

6.1.2.2.25 TRichViewEdit.ChangeStyleTemplates

Updates the document after editing style templates.

procedure `ChangeStyleTemplates(NewStyleTemplates: TRVStyleTemplateCollection688);`

(introduced in version 14)

The code below shows how to edit style templates for RichViewEdit1: TRichViewEdit. It assumes that you have RVStyleTmp: TRVStyle⁶³⁰ component, not linked to any RichView.

```
RVStyleTmp.Units655 := RichViewEdit1.Style.Units;
RVStyleTmp.StyleTemplates652.AssignStyleTemplates690(
  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`⁶⁸⁹).

Method type:  editing-style.

 **ScaleRichView note:** in TScaleRichViewEdit, 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`⁴⁹³;
- `ApplyParaStyleTemplate`⁴⁹¹;
- `ApplyTextStyleTemplate`⁴⁹⁴.

6.1.2.2.26 TRichViewEdit.Clear

Clears the document.

```
procedure Clear; override;
```

This method:

- deletes all content of the document (see `RichView.Clear`⁽²⁷⁹⁾);
- sets the current text style index (`CurTextStyleNo`⁽⁴⁷⁴⁾) and the current paragraph style index (`CurParaStyleNo`⁽⁴⁷³⁾) to zeros;
- clears undo and redo buffers (see `ClearUndo`⁽⁵⁰¹⁾);
- sets `Modified`⁽⁴⁷⁹⁾ to *False*.

This method does not check `ReadOnly`⁽⁴⁸⁰⁾, does not reformat and repaint component, does not generate `OnChange`⁽⁵⁷²⁾.

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`⁽²²⁵⁾ and `ClearRight`⁽²²⁶⁾ property for the first selected paragraph.

Method type:  editing-style.

See also:

- `TRVVAAlign`⁽¹⁰³³⁾ 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`⁽¹¹⁵⁾.

6.1.2.2.29 TRichViewEdit.ConvertToHotPicture

Converts the item type from picture⁽¹⁶³⁾ to *hot-picture*⁽¹⁶⁶⁾.

```
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`⁽²³⁷⁾-1) it works with the specified item

If this item is not a picture (`GetItemStyle`⁽³⁰²⁾/`CurItemStyle`⁽⁴⁷³⁾ <> *rvsPicture*), this method does nothing.

Method type:  editing-style.

See also:

- ConvertToPicture⁽⁵⁰²⁾.

6.1.2.2.30 TRichViewEdit.ConvertToPicture

Converts the item type from *hot-picture*⁽¹⁶⁶⁾ to *picture*⁽¹⁶³⁾

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⁽²³⁷⁾-1) it works with the specified item.

If this item is not a *hot-picture* (GetItemStyle⁽³⁰²⁾/CurlItemStyle⁽⁴⁷³⁾ <> *rvsHotPicture*), this method does nothing.

Method type:  editing-style.

See also:

- ConvertToHotPicture⁽⁵⁰¹⁾.

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⁽²¹⁸⁾ (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⁽²⁸¹⁾ method. If the selection was copied (at least one of copy formats is selected in Options⁽²⁴⁰⁾), it calls DeleteSelection⁽⁵⁰³⁾.

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⁽²⁸¹⁾;
- Paste⁽⁵³⁴⁾;
- SelectionExists⁽³⁴⁹⁾.

See also:

- Working with selection⁽¹⁰⁷⁾;
- Working with Clipboard⁽¹⁰⁸⁾.

6.1.2.2.33 TRichViewEdit.DeleteSelection

Deletes the selected fragment in the editor.

```
procedure DeleteSelection;
```

Method type: **I** editing-style.

See also:

- Working with selection ⁽¹⁰⁷⁾.

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 (998));
```

Parameters:

ItemNo – index of item which was modified.

ModifyData – the value obtained in BeginItemModify ⁽⁴⁹⁵⁾

Usually this function is used when modifying tables ⁽⁸⁴⁶⁾.

Example is in Table Operations Overview ⁽¹⁹⁹⁾

6.1.2.2.36 TRichViewEdit.EndUpdate

Reenables editor repainting that was turned off with the BeginUpdate ⁽⁴⁹⁸⁾ method. Repaints the editor.

```
procedure EndUpdate;
```

(meaning of this method is changed since v10)

Calls to BeginUpdate are cumulative, so calling EndUpdate will reenables 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 ⁽⁵⁰⁶⁾.

Caret Position	GetCheckpointAtCaret	GetCurrentCheckpoint ⁽⁵⁰⁶⁾
In the middle of text item	returns <i>nil</i>	returns <i>checkpoint</i> for this item
At the beginning of item and at the beginning of line	returns <i>checkpoint</i> for this item	returns <i>checkpoint</i> for this item
At the beginning of item and in the middle of line	returns <i>checkpoint</i> for this item	returns <i>checkpoint</i> for the previous item
At the end of item and at the end of line	returns <i>nil</i>	returns <i>checkpoint</i> for this item

This method is used in conjunction with `InsertCheckpoint(516)` and `RemoveCheckpointAtCaret(541)`. The alternative set of methods is: `SetCurrentCheckpointInfo(553)`, `GetCurrentCheckpoint(506)` and `RemoveCurrentCheckpoint(541)`.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.2.2.38 TRichViewEdit.GetCurrentBreakInfo

Returns main properties for the item of *break⁽¹⁶⁷⁾* type at the position of caret.

```
procedure GetCurrentBreakInfo(out AWidth: TRVStyleLength(1027);
  out AStyle: TRVBreakStyle(995); out AColor: TRVColor(996);
  out ATag: TRVTag(1029));
```

`GetCurrentBreakInfo(...)` is equivalent to `TopLevelEditor(481).GetBreakInfo(289)(TopLevelEditor(481).CurlItemNo(472), ...)`.

The item must be of *break⁽¹⁶⁷⁾* type (*rvsBreak⁽¹⁰⁵⁹⁾*), otherwise the method raises `ERichViewError(957)` exception. Type of item at the position of caret is returned by `CurlItemStyle(473)`.

Output parameters:

AWidth – line width (or rectangle height). This value is measured in `Style(253).Units(655)`.

AStyle – visual style of this *break*, see `TRVBreakStyle(995)` for possible values.

AColor – line color. If it is equal to `rvcNone(1038)`, `Style(253).TextStyles(654)[0].Color(701)` is used.

ATag – *tag* of the item. Use `SetCurrentTag(560)` or `SetCurrentBreakInfo(551)` to change *tag* of item as an editing operation.

Instead of this method, you can use `GetCurrentItemExtraIntPropertyEx(510)` and `GetCurrentTag(514)` 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(510)` and `GetCurrentItemExtraStrProperty(510)`.

See also methods:

- `SetCurrentBreakInfo`⁽⁵⁵¹⁾ (changes properties of *break* at the position of caret, as an editing operation);
- `GetBreakInfo`⁽²⁸⁹⁾ (returns properties of the specified *break*).

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾;
- `CurlItemStyle`⁽⁴⁷³⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.39 TRichViewEdit.GetCurrentBulletInfo

Returns main properties for the item of *bullet*⁽¹⁷⁰⁾ type at the position of caret.

VCL and LCL:

```
procedure GetCurrentBulletInfo(out AName: TRVUnicodeString(1032);
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag(1029));
```

FireMonkey:

```
procedure GetCurrentBulletInfo(out AName: TRVUnicodeString(1032);
  out AImageIndex: Integer; out AImageList: TCustomImageList;
  out ATag: TRVTag(1029);
  out AImageWidth, AImageHeight: TRVStyleLength(1027));
```

`GetCurrentBulletInfo(...)` is equivalent to `TopLevelEditor`⁽⁴⁸¹⁾.`GetBulletInfo`⁽²⁸⁹⁾ (`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾, ...).

Output parameters:

AName – name of *bullet*⁽¹⁷⁰⁾. It can also be read using `GetCurrentItemText`⁽⁵¹¹⁾ method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using `GetCurrentItemExtraIntPropertyEx`⁽⁵¹⁰⁾ method.

ATag – *tag* of the item. Use `SetCurrentTag`⁽⁵⁶⁰⁾ or `SetCurrentBulletInfo`⁽⁵⁵²⁾ to change *tag* of item as an editing operation. This value can also be read using `GetCurrentTag`⁽⁵¹⁴⁾ 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`⁽⁵¹⁰⁾ and `GetCurrentItemExtraStrProperty`⁽⁵¹⁰⁾.

See also methods:

- `SetCurrentBulletInfo`⁽⁵⁵²⁾ (changes properties of *bullet* at the position of caret, as an editing operation);
- `GetBulletInfo`⁽²⁸⁹⁾ (returns properties of the specified *bullet*).

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾;
- `CurlItemStyle`⁽⁴⁷³⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

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(993);
```

The same as `TopLevelEditor`⁽⁴⁸¹⁾.`GetItemCheckpoint`⁽²⁹⁸⁾ (`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾).

This method must be called only when the document is formatted.

See also methods:

- `SetCurrentCheckpointInfo`⁽⁵⁵³⁾ (changes *checkpoint* of item at the position of caret, as an editing operation);
- `GetItemCheckpoint`⁽²⁹⁸⁾ (returns *checkpoint* of the specified item).

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.2.2.41 TRichViewEdit.GetCurrentControlInfo

Returns main properties for the item of inserted control⁽¹⁶⁸⁾ type at position of caret.

```
procedure GetCurrentControlInfo(out AName: TRVUnicodeString(1032);  
  out ACtrl: TControl; out AVAlign: TRVVAlign(1033);  
  out ATag: TRVTag(1029));
```

(changed in version 18)

`GetCurrentControlInfo(...)` is equivalent to `TopLevelEditor`⁽⁴⁸¹⁾.`GetControlInfo`⁽²⁹³⁾ (`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾, ...).

Output parameters:

AName – name of control item⁽¹⁶⁸⁾. Do not confuse with **ACtrl.Name** property! This value can also be read using `GetCurrentItemText`⁽⁵¹¹⁾ method.

ACtrl – 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 **Ctrl.Tag** property!. Use **SetCurrentTag**⁽⁵⁶⁰⁾ or **SetCurrentControlInfo**⁽⁵⁵⁴⁾ to change *tag* of item as an editing operation. This value can also be read using **GetCurrentTag**⁽⁵¹⁴⁾ 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**⁽⁵¹⁰⁾ and **GetCurrentItemExtraStrProperty**⁽⁵¹⁰⁾.

See also methods:

- **SetCurrentControlInfo**⁽⁵⁵⁴⁾ (changes properties of control item at the position of caret, as an editing operation);
- **GetControlInfo**⁽²⁹³⁾ (returns properties of the specified control item).

See also properties:

- **CurlItemNo**⁽⁴⁷²⁾;
- **CurlItemStyle**⁽⁴⁷³⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.42 TRichViewEdit.GetCurrentHotspotInfo

Returns main properties for the item of *hotspot*⁽¹⁷²⁾ type at the position of caret.

VCL and LCL:

```
procedure GetCurrentHotspotInfo(out AName: TRVUnicodeString(1032);
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag(1029));
```

(changed in version 18)

FireMonkey:

```
procedure GetCurrentHotspotInfo(out AName: TRVUnicodeString(1032);
  out AImageIndex, AHotImageIndex: Integer;
  out AImageList: TCustomImageList;
  out ATag: TRVTag(1029);
  out AImageWidth, AImageHeight: TRVStyleLength(1027));
```

GetCurrentHotspotInfo(...) is equivalent to **TopLevelEditor**⁽⁴⁸¹⁾.**GetHotspotInfo**⁽²⁹⁵⁾ (**TopLevelEditor**⁽⁴⁸¹⁾.**CurlItemNo**⁽⁴⁷²⁾, ...).

Output parameters:

AName – name of *hotspot*⁽¹⁷²⁾. It can also be read using **GetCurrentItemText**⁽⁵¹¹⁾ 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**⁽⁵¹⁰⁾ method.

AHotImageIndex – index in **AlmageList** 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`⁽⁵¹⁰⁾ method.

ATag – *tag* of the item. Use `SetCurrentTag`⁽⁵⁶⁰⁾ or `SetCurrentHotspotInfo`⁽⁵⁵⁵⁾ to change *tag* of item as an editing operation. This value can also be read using `GetCurrentTag`⁽⁵¹⁴⁾ 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`⁽⁵¹⁰⁾ and `GetCurrentItemExtraStrProperty`⁽⁵¹⁰⁾.

See also methods:

- `SetCurrentHotspotInfo`⁽⁵⁵⁵⁾ (changes properties of *hotspot* at the position of caret, as an editing operation);
- `GetHotspotInfo`⁽²⁹⁵⁾ (returns properties of the specified *hotspot*).

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾;
- `CurlItemStyle`⁽⁴⁷³⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.43 TRichViewEdit.GetCurrentItem

Returns object representing item at position of caret.

function `GetCurrentItem`: TCustomRVItemInfo⁽⁸⁴⁴⁾;

(introduced in version 1.4)

This method is usually used when working with tables⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, references to notes⁽¹⁸⁴⁾.

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`⁽⁵⁰⁹⁾ instead of `GetCurrentItem`.

This method must be called only when the document is formatted.

When called for an empty editor (i.e. `ItemCount`⁽²³⁷⁾=0), the method returns *nil*.

See also methods:

- `GetCurrentItemEx`⁽⁵⁰⁹⁾;
- `InsertItem`⁽⁵²²⁾;
- `GetItem`⁽²⁹⁶⁾.

See also properties:

- `CurlItemNo` ⁽⁴⁷²⁾;
- `CurlItemStyle` ⁽⁴⁷³⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾;
- Tables in RichView ⁽¹⁹⁰⁾.

6.1.2.2.44 TRichViewEdit.GetCurrentItemEx

Returns object at the position of caret representing item of the given class.

```
function GetCurrentItemEx(RequiredClass: TCustomRVItemInfoClass (844);  
  out ItemRichViewEdit: TCustomRichViewEdit;  
  out Item: TCustomRVItemInfo (844)): Boolean;
```

(introduced in version 1.4)

This method is usually used when working with tables ⁽¹⁹⁰⁾.

Input parameter:

RequiredClass – the required class of item, set it to `TRVTableItemInfo` ⁽⁸⁴⁶⁾.

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` ⁽⁵⁰⁸⁾;
- `InsertItem` ⁽⁵²²⁾;
- `GetItem` ⁽²⁹⁶⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾;
- Tables in RichView ⁽¹⁹⁰⁾.

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: TRVExtraItemProperty1000; out Value: Integer): Boolean;
function GetCurrentItemExtraIntPropertyEx(
  Prop: Integer; out Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

GetCurrentItemExtraIntProperty[Ex](...) is equivalent to `TopLevelEditor481.GetItemExtraIntProperty[Ex]300(TopLevelEditor481.CurlItemNo472, ...)`.

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraIntProperty**, **Prop**'s type is `TRVExtraItemProperty1000`. 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***` constants¹⁰⁵¹

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]556` (set a value of an integer property for the item at the position of caret, as an editing operation);
- `SetItemExtraIntProperty[Ex]Ed562` (set a value of an integer property for the specified item, as an editing operation);
- `GetCurrentItemExtraStrProperty510` (return a value of the specified *string* property of the item at the position of caret).

See also methods of TRichView

- `GetItemExtraIntProperty[Ex]300`.

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: TRVExtraItemStrProperty1005;
  out Value: TRVUnicodeString1032): Boolean;
function GetCurrentItemExtraStrProperty(
  Prop: Integer; out Value: TRVUnicodeString1032): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

GetCurrentItemExtraStrProperty[Ex](...) is equivalent to `TopLevelEditor481.GetItemExtraStrProperty[Ex]300(TopLevelEditor481.CurlItemNo472, ...)`.

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty⁽¹⁰⁰⁵⁾. 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⁽¹⁰⁵⁶⁾

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]**⁽⁵⁵⁷⁾ (set a value of a string property for the item at the position of caret, as an editing operation);
- **SetItemExtraStrProperty[Ex]Ed**⁽⁵⁶²⁾ (set a value of a string property for the specified item, as an editing operation);
- **GetCurrentItemExtraIntProperty[Ex]**⁽⁵¹⁰⁾ (return a value of the specified *integer* property of the item at the position of caret).

See also methods of TRichView:

- **GetItemExtraStrProperty[Ex]**⁽³⁰⁰⁾.

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(993);
function GetCurrentItemTextW: TRVUnicodeString(1032);
```

(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**⁽⁴⁸¹⁾.**GetItemText**⁽³⁰³⁾ (**TopLevelEditor**⁽⁴⁸¹⁾.CurItemNo⁽⁴⁷²⁾).
- **GetCurrentItemTextA** is the same as **TopLevelEditor**⁽⁴⁸¹⁾.**GetItemTextA**⁽³⁰³⁾ (**TopLevelEditor**⁽⁴⁸¹⁾.CurItemNo⁽⁴⁷²⁾).
- **GetCurrentItemTextW** is the same as **TopLevelEditor**⁽⁴⁸¹⁾.**GetItemTextW**⁽³⁰³⁾ (**TopLevelEditor**⁽⁴⁸¹⁾.CurItemNo⁽⁴⁷²⁾).

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**⁽⁷⁰⁰⁾. If it is equal to **DEFAULT_CHARSET**, and for non-text items, **Style**⁽²⁵³⁾.**DefCodePage**⁽⁶³⁵⁾ 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⁽¹⁵⁴⁾

These methods must be called only when the document is formatted.

See also methods:

- `SetCurrentItemText -A -W` ⁽⁵⁵⁷⁾.

See also methods of TCustomRichView:

- `GetItemText -A -W` ⁽³⁰³⁾.

See also properties:

- `CurlItemNo` ⁽⁴⁷²⁾;
- `CurlItemStyle` ⁽⁴⁷³⁾.

See also:

- Obtaining RichView items ⁽¹¹¹⁾;
- Item types ⁽⁸⁵⁾;
- "Tags" ⁽⁹¹⁾;
- Unicode in RichView ⁽¹³⁰⁾.

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` ⁽¹⁰³³⁾;

(introduced in version 12)

The same as `TopLevelEditor` ⁽⁴⁸¹⁾.`GetItemVAlign` ⁽³⁰³⁾(`TopLevelEditor` ⁽⁴⁸¹⁾.`CurlItemNo` ⁽⁴⁷²⁾).

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 ⁽¹⁶¹⁾, tables ⁽¹⁷³⁾, *breaks* ⁽¹⁶⁷⁾, tabs ⁽¹⁶²⁾, list markers ⁽¹⁷⁴⁾, footnotes ⁽¹⁸⁰⁾ and endnotes ⁽¹⁷⁸⁾.

See also methods:

- `SetCurrentItemVAlign` ⁽⁵⁵⁸⁾.

See also properties:

- `CurlItemNo` ⁽⁴⁷²⁾.

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`⁽³⁰⁶⁾.

6.1.2.2.50 TRichViewEdit.GetCurrentMisspelling

Returns misspelled word at the position of caret, optionally selects it.

```
function GetCurrentMisspelling(SelectIt: Boolean;
  out Word: TRVUnicodeString1032; 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⁽⁶⁵⁴⁾ collection of the linked TRVStyle component)⁽²⁵³⁾.

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⁽¹³²⁾.

6.1.2.2.51 TRichViewEdit.GetCurrentPictureInfo

Returns main properties for the item of picture⁽¹⁶³⁾/*hot-picture*⁽¹⁶⁶⁾ type at the position of caret.

```
procedure GetCurrentPictureInfo(out AName: TRVUnicodeString1032;
  out Agr: TRVGraphic970; out AVAlign: TRVValign1033; out ATag: TRVTag1029);
```

(changed in version 18)

GetCurrentPictureInfo(...) is equivalent to TopLevelEditor⁽⁴⁸¹⁾.**GetPictureInfo**⁽³¹⁰⁾(TopLevelEditor⁽⁴⁸¹⁾.CurlItemNo⁽⁴⁷²⁾, ...).

Output parameters:

AName – name of the item. It can also be read using GetCurrentItemText⁽⁵¹¹⁾ 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⁽⁵⁶⁰⁾ or SetCurrentPictureInfo⁽⁵⁵⁹⁾ to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag⁽⁵¹⁴⁾ 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`⁽⁵¹⁰⁾ and `GetCurrentItemExtraStrProperty`⁽⁵¹⁰⁾.

See also methods:

- `SetCurrentPictureInfo`⁽⁵⁵⁹⁾ (changes properties of picture at the position of caret, as an editing operation);
- `GetPictureInfo`⁽³¹⁰⁾ (returns properties of the specified picture).

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾;
- `CurlItemStyle`⁽⁴⁷³⁾.

See also:

- Obtaining RichView items⁽¹¹¹⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.52 TRichViewEdit.GetCurrentTag

Returns tag of the item at the caret position

```
function GetCurrentTag: TRVTag(1029);
```

The same as `TopLevelEditor`⁽⁴⁸¹⁾.`GetItemTag`⁽³⁰²⁾(`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾).

Use `SetCurrentTag`⁽⁵⁶⁰⁾ 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`⁽⁵⁶⁰⁾.

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾.

See also

- "Tags"⁽⁹¹⁾.

6.1.2.2.53 TRichViewEdit.GetCurrentTextInfo

Returns text and *tag* for text item at the position of caret

```
procedure GetCurrentTextInfo(out AText: TRVUnicodeString(1032);  
out ATag: TRVTag(1029));
```

`GetCurrentTextInfo(...)` is equivalent to `TopLevelEditor`⁽⁴⁸¹⁾.`GetTextInfo`⁽³¹³⁾(`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾, ...).

Use `SetCurrentTag`⁽⁵⁶⁰⁾ 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` ⁽⁴⁷²⁾;
- `CurItemStyle` ⁽⁴⁷³⁾.

See also:

- Obtaining items ⁽¹¹¹⁾;
- Item types ⁽⁸⁵⁾;
- "Tags" ⁽⁹¹⁾.

6.1.2.2.54 TRichViewEdit.InsertBreak

Inserts *break* ⁽¹⁶⁷⁾ (horizontal line) at the position of caret.

```
function InsertBreak(AWidth: TRVStyleLength (1027); AStyle: TRVBreakStyle (995);  
    AColor: TRVColor (996)): Boolean;
```

Parameters:

AWidth – width of the line, measured in `Style` ⁽²⁵³⁾.Units ⁽⁶⁵⁵⁾;

AStyle – break type;

AColor – color of line. If it is equal to `rvclNone` ⁽¹⁰³⁸⁾, the *break* has the color of the 0th text style ⁽⁶⁵⁴⁾.

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` ⁽²⁶²⁾.

See also:

- Inserting items at position of caret ⁽¹¹⁴⁾;
- Item types ⁽⁸⁵⁾.

6.1.2.2.55 TRichViewEdit.InsertBullet

Inserts *bullet* ⁽¹⁷⁰⁾ at the position of caret.

VCL and LCL:

```
function InsertBullet(ImageIndex: Integer;  
    ImageList: TCustomImageList): Boolean;
```

FireMonkey:

```
function InsertBullet(ImageIndex: Integer;  
    ImageList: TCustomImageList;  
    ImageWidth, ImageHeight: TRVStyleLength (1027)): 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⁽²⁶³⁾;

See also:

- Inserting items at position of caret⁽¹¹⁴⁾;
- Item types⁽⁸⁵⁾.

6.1.2.2.56 TRichViewEdit.InsertCheckpoint

Inserts *checkpoint* at the position of caret.

```
function InsertCheckpoint(const ATag: TRVTag(1029);
  const AName: TRVUnicodeString(1032);
  ARaiseEvent: Boolean): Boolean;
```

(introduced in version 10; changed in versions 18 and 19)

This method is similar to SetCurrentCheckpointInfo⁽⁵⁵³⁾, but inserts *checkpoint* exactly at the position of caret.

If the caret is at the beginning of item, this methods calls SetCheckpointInfoEd⁽⁵⁴⁹⁾ to add *checkpoint* for this item.

If the caret is at the middle of text item⁽¹⁶¹⁾, this method splits this item at the position of caret, and calls SetCheckpointInfoEd⁽⁵⁴⁹⁾ 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⁽⁵⁴⁹⁾ to add *checkpoint* for the first paragraph item (or the second item, if the first one is list marker⁽¹⁷⁴⁾).

If the caret is at the end of document, this methods does nothing.

This method is used in conjunction with GetCheckpointAtCaret⁽⁵⁰³⁾ and RemoveCheckpointAtCaret⁽⁵⁴¹⁾ (GetCurrentCheckpoint⁽⁵⁰⁶⁾ and RemoveCurrentCheckpoint⁽⁵⁴¹⁾ 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⁽³⁷²⁾ 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⁽¹¹²⁾;
- "Checkpoints"⁽⁸⁷⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.57 TRichViewEdit.InsertControl

Inserts Delphi/C++Builder control⁽¹⁶⁸⁾ at the position of caret.

```
function InsertControl(const Name: TRVUnicodeString(1032); ctrl: TControl;  
    VAlign: TRVVAlign(1033)): 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⁽¹⁰³³⁾ 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⁽²⁶⁵⁾;

See also:

- Inserting items at position of caret ⁽¹¹⁴⁾;
- Item types ⁽⁸⁵⁾.

6.1.2.2.58 TRichViewEdit.InsertDocXFromFileEd

Inserts DocX (Microsoft Word Document) from file **FileName** at the position of caret.

```
function InsertDocXFromFileEd(
  const FileName: TRVUnicodeString(1032)): Boolean;
```

(Introduced in version 19)

Parameters for loading are in TRichView.RTFReadProperties ⁽²⁴⁶⁾. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode ⁽⁴⁵⁸⁾ and ParaStyleMode ⁽⁴⁵⁶⁾. 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 ⁽²¹⁸⁾.

Method type:  editing-style method for insertion.

If style templates are used ⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode ⁽²⁵⁴⁾ <>*rvstimIgnoreSourceStyleTemplates*, the method merges a DocX style sheet into Style ⁽²⁵³⁾.StyleTemplates ⁽⁶⁵²⁾, and reads text and paragraph styles according to StyleTemplateInsertMode ⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange ⁽⁴¹⁰⁾ event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode ⁽⁴⁵⁴⁾)

See also methods of TRichViewEdit:

- InsertDocXFromStreamEd ⁽⁵¹⁸⁾.

See also methods of TRichView:

- LoadDocX ⁽³¹⁸⁾.

See also events of TRichView:

- OnImportPicture ⁽³⁷⁸⁾;
- OnReadHyperlink ⁽³⁸⁸⁾;
- OnStyleTemplatesChange ⁽⁴¹⁰⁾,

See also:

- Saving and loading ⁽¹²²⁾;
- Inserting items at position of caret ⁽¹¹⁴⁾.

6.1.2.2.59 TRichViewEdit.InsertDocXFromStreamEd

Inserts DocX (Microsoft Word Document) from the **Stream** at the position of caret.

```
function InsertDocXFromStreamEd(Stream: TStream): Boolean;
```

(Introduced in version 19)

Parameters for loading are in TRichView.RTFReadProperties ⁽²⁴⁶⁾. The most important properties affecting DocX loading are RTFReadProperties.TextStyleMode ⁽⁴⁵⁸⁾ and ParaStyleMode ⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

Method type:  editing-style method for insertion.

If style templates are used⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rversAddIfNeeded*, and StyleTemplateInsertMode⁽²⁵⁴⁾ <>*rvstimIgnoreSourceStyleTemplates*, the method merges a DocX style sheet into Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾, and reads text and paragraph styles according to StyleTemplateInsertMode⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange⁽⁴¹⁰⁾ event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.DocXErrorCode⁽⁴⁵⁴⁾)

See also methods of TRichViewEdit:

- InsertDocXFromFileEd⁽⁵¹⁸⁾.

See also methods of TRichView:

- LoadDocXFromStream⁽³¹⁹⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾;
- OnStyleTemplatesChange⁽⁴¹⁰⁾.

See also:

- Saving and loading⁽¹²²⁾;
- Inserting items at position of caret⁽¹¹⁴⁾.

6.1.2.2.60 TRichViewEdit.InsertHotPicture

Inserts picture with hypertext link⁽¹⁶⁶⁾ at the position of caret.

```
function InsertHotPicture(const Name: TRVUnicodeString(1032);
  gr: TRVGraphic(970); VAlign: TRVVAlign(1033);
  ImageWidth: TRVStyleLength(1027) = 0;
  ImageHeight: TRVStyleLength(1027) = 0: Boolean;
```

(introduced in version 1.5; changed in version 21)

The same as InsertPicture⁽⁵²⁶⁾, 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⁽²⁶⁷⁾.

See also:

- Inserting items at position of caret ⁽¹¹⁴⁾;
- Hypertext in RichView ⁽⁹²⁾;
- Item types ⁽⁸⁵⁾.

6.1.2.2.61 TRichViewEdit.InsertHotspot

Inserts *hotspot* ⁽¹⁷²⁾ at the position of caret.

VCL and LCL:

```
function InsertHotspot(ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList): Boolean;
```

FireMonkey:

```
function InsertHotspot(ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList;
  ImageWidth, ImageHeight: TRVStyleLength (1027)): 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 ⁽²⁶⁸⁾.

See also:

- Inserting items at position of caret ⁽¹¹⁴⁾;
- Hypertext in RichView ⁽⁹²⁾;
- Item types ⁽⁸⁵⁾.

6.1.2.2.62 TRichViewEdit.InsertHTMLFromFileEd

Inserts HTML from file **FileName** at the position of caret.

```
function InsertHTMLFromFileEd(  
    const FileName: TRVUnicodeString1032;  
    CodePage: TRVCodePage996 = 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.HTMLReadProperties²³⁷.

If style templates are used²⁵⁶, and StyleTemplateInsertMode²⁵⁴ `<>rvstimgnoreSourceStyleTemplates`, the method merges a HTML style sheet into Style²⁵³.StyleTemplates⁶⁵², and reads text and paragraph styles according to StyleTemplateInsertMode²⁵⁴. The method calls OnStyleTemplatesChange⁴¹⁰ event.

See details about HTML style sheet and encoding detection in the topic about LoadHTML³²².

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertHTMLFromStreamEd⁵²¹;
- PasteHTML⁵³⁶.

See also methods of TRichView:

- LoadHTML³²².

See also events of TRichView:

- OnImportPicture³⁷⁸;
- OnImportFile³⁷⁸;
- OnReadHyperlink³⁸⁸;
- OnStyleTemplatesChange⁴¹⁰,

See also:

- Saving and loading¹²²;
- Inserting items at position of caret¹¹⁴.

6.1.2.2.63 TRichViewEdit.InsertHTMLFromStreamEd

Inserts HTML from file **FileName** at the position of caret.

```
function InsertHTMLFromFileEd(  
    const FileName: TRVUnicodeString1032;  
    CodePage: TRVCodePage996 = CP_ACP): Boolean;
```

(Introduced in version 21)

Parameters

Stream – stream containing HTML

Path defines a base path for images. If `HTMLReadProperties(237).BasePathLinks(425) = 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(237)`.

If style templates are used⁽²⁵⁶⁾, and `StyleTemplateInsertMode(254) <>rvstimIgnoreSourceStyleTemplates`, the method merges a HTML style sheet into `Style(253).StyleTemplates(652)`, and reads text and paragraph styles according to `StyleTemplateInsertMode(254)`. The method calls `OnStyleTemplatesChange(410)` event.

See details about HTML style sheet and encoding detection in the topic about `LoadHTML(322)`.

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- `InsertHTMLFromFileEd(521)`;
- `PasteHTML(536)`.

See also methods of TRichView:

- `LoadHTML(322)`.

See also events of TRichView:

- `OnImportPicture(378)`;
- `OnImportFile(378)`;
- `OnReadHyperlink(388)`;
- `OnStyleTemplatesChange(410)`,

See also:

- `Saving and loading(122)`;
- `Inserting items at position of caret(114)`.

6.1.2.2.64 TRichViewEdit.InsertItem

Inserts item at the position of caret.

```
function InsertItem(const Name: TRVUnicodeString(1032);
  Item: TCustomRVItemInfo(844)): 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⁽¹⁷³⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, references to notes⁽¹⁸⁴⁾, page numbers⁽¹⁸⁵⁾, page counts⁽¹⁸⁶⁾, equations⁽¹⁸⁷⁾ (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.ParaStyles⁶⁴⁸).

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-only⁴⁸⁰ or protected⁷⁰⁵, or when inserting in table having multicell selection); in this case **Item** was destroyed (freed) by InsertItem.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also:

- Inserting items at position of caret¹¹⁴;
- Item types⁸⁵.

6.1.2.2.65 TRichViewEdit.InsertMarkdownFromFileEd

Inserts **Markdown** from file **FileName** at the position of caret.

```
function InsertMarkdownFromFileEd(const FileName: TRVUnicodeString1032;  
    CodePage: TRVCodePage996 = 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 MarkdownProperties²³⁹. See the topic about the class of this property⁴³⁹ 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 tags⁹¹. You can customize loading using OnReadHyperlink³⁸⁸ event.

Method type:  editing-style method for insertion.

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertMarkdownFromStreamEd⁵²⁴.

See also methods of TRichView:

- LoadMarkdown⁽³²⁵⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾.

See also:

- Saving and loading⁽¹²²⁾;
- Inserting items at position of caret⁽¹¹⁴⁾.

6.1.2.2.66 TRichViewEdit.InsertMarkdownFromStreamEd

Inserts **Markdown** from stream **Stream** at the position of caret.

```
function InsertMarkdownFromStreamEd(Stream: TStream;
  const Path: TRVUnicodeString(1032) = '';
  CodePage: TRVCodePage(996) = 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**⁽²³⁹⁾.**LoadOptions**⁽⁴⁴⁶⁾, this path is also used for hyperlinks.

Parameters for loading are in **MarkdownProperties**⁽²³⁹⁾. See the topic about the class of this property⁽⁴³⁹⁾ 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⁽⁹¹⁾. You can customize loading using **OnReadHyperlink**⁽³⁸⁸⁾ event.

Method type:  editing-style method for insertion.

Return value:

"successful insertion?"

See also methods of TRichViewEdit:

- InsertMarkdownFromFileEd⁽⁵²³⁾.

See also methods of TRichView:

- LoadMarkdownFromStream⁽³²⁵⁾.

See also events of TRichView:

- OnImportPicture⁽³⁷⁸⁾;
- OnReadHyperlink⁽³⁸⁸⁾.

See also:

- Saving and loading⁽¹²²⁾;
- Inserting items at position of caret⁽¹¹⁴⁾.

6.1.2.2.67 TRichViewEdit.InsertOEMTextFromFile

Loads text file (in OEM-defined character set) **FileName** and inserts its content at the position of caret.

```
function InsertOEMTextFromFile(const FileName: TRVUnicodeString1032): 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 InsertTextFromFile⁵³³.

Return value:

"Was reading from from file successful?"

See also methods:

- InsertTextFromFile⁵³³;
- InsertText⁵³².

See also properties:

- CurParaStyleNo⁴⁷³;
- CurTextStyleNo⁴⁷⁴.

See also properties of TRVStyle:

- TextStyles⁶⁵⁴;
- ParaStyles⁶⁴⁸;
- SpacesInTab⁶⁵².

See also:

- Saving and loading¹²²;
- Inserting items at position of caret¹¹⁴.

6.1.2.2.68 TRichViewEdit.InsertPageBreak

Inserts explicit page break at the position of caret.

```
procedure InsertPageBreak
```

(*introduced in version 1.2*)

The method deselects the document before performing its work.

If the caret is not at the beginning of some item which is always displayed from the new line (see IsFromNewLine³¹⁷), 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 PageBreaksBeforeItems²⁴⁴).

Method type:  editing-style.

See also:

- `RemoveCurrentPageBreak`⁽⁵⁴²⁾.

See also properties of RichView:

- `PageBreaksBeforeItems`⁽²⁴⁴⁾;
- `Options`⁽²⁴⁰⁾ (`rvoShowPageBreaks`).

See also properties of RVStyle:

- `PageBreakColor`⁽⁶⁴⁷⁾.

6.1.2.2.69 TRichViewEdit.InsertPicture

Inserts picture⁽¹⁶³⁾ at the position of caret.

```
function InsertPicture(const Name: TRVUnicodeString(1032); gr: TRVGraphic(970);
  VAlign: TRVVAlign(1033); ImageWidth: TRVStyleLength(1027) = 0;
  ImageHeight: TRVStyleLength(1027) = 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⁽¹⁰⁰⁰⁾, 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`⁽¹⁰⁴⁶⁾). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see `TRVVAlign`⁽¹⁰³³⁾ for possible values.

The image is displayed stretched to **ImageWidth** x **ImageHeight** (if they are non-zero, see `rvepImageWidth` and `rvepImageHeight` in `IDH_Type_TRVExtraltemProperty`⁽¹⁰⁰⁰⁾).

Method type:  editing-style method for insertion.

If you want this picture to be a hypertext link, use `InsertHotPicture`⁽⁵¹⁹⁾ 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: TRVGraphic970;
...
if OpenPictureDialog1.Execute then
  gr := RVGraphicHandler1057.LoadFromFile972(OpenPictureDialog1.FileName);
  if gr<>nil then
    MyRichViewEdit.InsertPicture('', gr, rvvaBaseline)
  else
    // process a picture loading error here
end;

```

See also methods of TCustomRichView:

- AddPicture²⁷².

See also:

- Inserting items at position of caret¹¹⁴;
- Hypertext in RichView⁹²;
- Item types⁸⁵.

6.1.2.2.70 TRichViewEdit.InsertRTFFromFileEd

Inserts RTF (Rich Text Format) from file **FileName** at the position of caret.

```

function InsertRTFFromFileEd(const FileName: TRVUnicodeString1032): Boolean;
(Introduced in version 1.5; changed in version 18)

```

Parameters for loading are in TRichView.RTFReadProperties²⁴⁶. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode⁴⁵⁸ and ParaStyleMode⁴⁵⁶. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor²¹⁸.

Method type:  editing-style method for insertion.

If style templates are used²⁵⁶, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=rvrsAddIfNeeded, and StyleTemplateInsertMode²⁵⁴<>rvstimIgnoreSourceStyleTemplates, the method merges an RTF style sheet into Style²⁵³.StyleTemplates⁶⁵², and reads text and paragraph styles according to StyleTemplateInsertMode²⁵⁴. The method calls OnStyleTemplatesChange⁴¹⁰ event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.RTFErrorCode⁴⁵⁷)

See also methods of TRichViewEdit:

- InsertRTFFromStreamEd⁵²⁸;
- PasteRTF⁵³⁷.

See also methods of TRichView:

- LoadRTF³²⁶.

See also events of TRichView:

- OnImportPicture³⁷⁸;
- OnReadHyperlink³⁸⁸;
- OnStyleTemplatesChange⁴¹⁰,

See also:

- Saving and loading ⁽¹²²⁾;
- Inserting items at position of caret ⁽¹¹⁴⁾.

6.1.2.2.71 TRichViewEdit.InsertRTFFromStreamEd

Inserts RTF (Rich Text Format) from the **Stream** at the position of caret.

```
function InsertRTFFromStreamEd(Stream: TStream): Boolean;
```

(Introduced in version 1.5)

Parameters for loading are in TRichView.RTFReadProperties ⁽²⁴⁶⁾. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode ⁽⁴⁵⁸⁾ and ParaStyleMode ⁽⁴⁵⁶⁾. 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 ⁽²¹⁸⁾.

Method type:  editing-style method for insertion.

If style templates are used ⁽²⁵⁶⁾, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode ⁽²⁵⁴⁾ <>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style ⁽²⁵³⁾.StyleTemplates ⁽⁶⁵²⁾, and reads text and paragraph styles according to StyleTemplateInsertMode ⁽²⁵⁴⁾. The method calls OnStyleTemplatesChange ⁽⁴¹⁰⁾ event.

Return value:

"successful insertion?" (extended information is in TRichView.RTFReadProperties.RTFErrorCode ⁽⁴⁵⁷⁾)

See also methods of TRichViewEdit:

- InsertRTFFromFileEd ⁽⁵²⁷⁾;
- PasteRTF ⁽⁵³⁷⁾.

See also methods of TRichView:

- LoadRTFFromStream ⁽³²⁷⁾.

See also events of TRichView:

- OnImportPicture ⁽³⁷⁸⁾;
- OnReadHyperlink ⁽³⁸⁸⁾;
- OnStyleTemplatesChange ⁽⁴¹⁰⁾.

See also:

- Saving and loading ⁽¹²²⁾;
- Inserting items at position of caret ⁽¹¹⁴⁾.

6.1.2.2.72 TRichViewEdit.InsertRVFFromFileEd

Inserts RVF (RichView Format ⁽¹²⁴⁾) from the file **FileName** at the position of caret.

```
function InsertRVFFromFileEd(const FileName: TRVUnicodeString (1032)): Boolean;
```

(changed in version 18)

Settings for RVF loading can be changed in the TRichView component editor ⁽²¹⁸⁾.

Method type:  editing-style method for insertion.

If style templates are used ⁽²⁵⁶⁾, and `RVFTextStylesReadMode` ⁽²⁵¹⁾ = `RVFParaStylesReadMode` ⁽²⁵¹⁾ = `rvf_sInsertMerge`, and `StyleTemplateInsertMode` ⁽²⁵⁴⁾ `<>rvstimIgnoreSourceStyleTemplates`, the method merges style templates from RVF into `Style` ⁽²⁵³⁾.`StyleTemplates` ⁽⁶⁵²⁾, and reads text and paragraph styles according to `StyleTemplateInsertMode` ⁽²⁵⁴⁾. The method calls `OnStyleTemplatesChange` ⁽⁴¹⁰⁾ event.

Return value: "successful insertion?"

See also methods of TRichViewEdit:

- `InsertRVFFromStreamEd` ⁽⁵²⁹⁾;
- `PasteRVF` ⁽⁵³⁸⁾.

See also methods of TRichView:

- `LoadRVF` ⁽³²⁸⁾.

▪ **See also properties:**

- `RVFOptions` ⁽²⁴⁷⁾;
- `RVFWarnings` ⁽²⁵¹⁾;
- `RVFTextStylesReadMode` ⁽²⁵¹⁾ (this method type: RVF insertion);
- `RVFParaStylesReadMode` ⁽²⁵¹⁾ (this method type: RVF insertion);
- `UseStyleTemplates` ⁽²⁵⁶⁾;
- `StyleTemplateInsertMode` ⁽²⁵⁴⁾.

See also events:

- `OnRVFImageListNeeded` ⁽³⁹³⁾;
- `OnRVFControlNeeded` ⁽³⁹²⁾;
- `OnRVFPictureNeeded` ⁽³⁹³⁾;
- `OnControlAction` ⁽³⁷³⁾ (`ControlAction=rvcaAfterRVFLoad`).

See also:

- `TRichView` component editor ⁽²¹⁸⁾;
- `Saving and loading` ⁽¹²²⁾;
- `Inserting items at position of caret` ⁽¹¹⁴⁾;
- `RVF` ⁽¹²⁴⁾.

6.1.2.2.73 TRichViewEdit.InsertRVFFromStreamEd

Inserts RVF (RichView Format ⁽¹²⁴⁾) from the **Stream** at the position of caret.

function `InsertRVFFromStreamEd(Stream: TStream): Boolean;`

Settings for RVF loading can be changed in the `TRichView` component editor ⁽²¹⁸⁾.

Method type:  editing-style method for insertion.

If style templates are used ⁽²⁵⁶⁾, and `RVFTextStylesReadMode` ⁽²⁵¹⁾ = `RVFParaStylesReadMode` ⁽²⁵¹⁾ = `rvf_sInsertMerge`, and `StyleTemplateInsertMode` ⁽²⁵⁴⁾ `<>rvstimIgnoreSourceStyleTemplates`, the method merges style templates from RVF into `Style` ⁽²⁵³⁾.`StyleTemplates` ⁽⁶⁵²⁾, and reads text and paragraph styles according to `StyleTemplateInsertMode` ⁽²⁵⁴⁾. The method calls `OnStyleTemplatesChange` ⁽⁴¹⁰⁾ event.

Return value: "successful insertion?"

See also methods of TRichViewEdit:

- InsertRVFFromFileEd⁽⁵²⁸⁾;
- PasteRVF⁽⁵³⁸⁾.

See also methods of TRichView:

- LoadRVFFromStream⁽³²⁷⁾;
- InsertRVFFromStream⁽³¹⁶⁾;
- AppendRVFFromStream⁽²⁷⁶⁾.
- **See also properties:**
- RVFOptions⁽²⁴⁷⁾;
- RVFWarnings⁽²⁵¹⁾;
- RVFTextStylesReadMode⁽²⁵¹⁾ (this method type: RVF insertion);
- RVFParaStylesReadMode⁽²⁵¹⁾ (this method type: RVF insertion);
- UseStyleTemplates⁽²⁵⁶⁾;
- StyleTemplateInsertMode⁽²⁵⁴⁾.

See also events:

- OnRVFImageListNeeded⁽³⁹³⁾;
- OnRVFControlNeeded⁽³⁹²⁾;
- OnRVFPictureNeeded⁽³⁹³⁾;
- OnControlAction⁽³⁷³⁾ (ControlAction=rvcaAfterRVFLoad);
- OnStyleTemplatesChange⁽⁴¹⁰⁾.

See also:

- TRichView component editor⁽²¹⁸⁾;
- Saving and loading⁽¹²²⁾;
- Inserting items at position of caret⁽¹¹⁴⁾;
- RVF⁽¹²⁴⁾.

6.1.2.2.74 TRichViewEdit.InsertStringTag -A -W

The methods Insert text string **s** with associated **Tag** at the position of caret.

```
function InsertStringTag(const s: String;
  const Tag: TRVTag(1029)): Boolean;
function InsertStringATag(const s: TRVAnsiString(993);
  const Tag: TRVTag(1029)): Boolean;
function InsertStringWTag(const s: TRVUnicodeString(1032);
  const Tag: TRVTag(1029)): Boolean;
```

(introduced in version 1.6)

Inserted text (one text item) has the current text⁽⁴⁷⁴⁾ and current paragraph⁽⁴⁷³⁾ style.

Methods type: **I** editing-style method for insertion.

Unlike InsertText⁽⁵³²⁾:

- **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⁽⁶⁵²⁾=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`⁽⁵³²⁾ when possible.

Unicode note:

Internally, text is stored as Unicode. `InsertStringATag` converts ANSI to Unicode using `Style`⁽²⁵³⁾.`DefCodePage`⁽⁶³⁵⁾.

`InsertStringTag` works:

- like `InsertStringATag`, in Delphi 2007 and older
- like `InsertStringWTag`, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus⁽¹⁵⁴⁾

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`⁽⁵³²⁾.

See also properties:

- `CurParaStyleNo`⁽⁴⁷³⁾;
- `CurTextStyleNo`⁽⁴⁷⁴⁾.

See also properties of TRVStyle:

- `TextStyles`⁽⁶⁵⁴⁾;
- `ParaStyles`⁽⁶⁴⁸⁾;
- `SpacesInTab`⁽⁶⁵²⁾.

See also:

- Inserting items at position of caret⁽¹¹⁴⁾;
- `Tags`⁽⁹¹⁾.

6.1.2.2.75 TRichViewEdit.InsertTab

Inserts tabulator⁽¹⁶²⁾ at the position of caret.

function `InsertTab`: `Boolean`;

(introduced in v1.9)

Inserted tabulator has the current text⁽⁴⁷⁴⁾ and the current paragraph⁽⁴⁷³⁾ style.

This method always inserts a tabulator, regardless of `RVStyle.SpacesInTab`⁽⁶⁵²⁾ mode.

If `RVStyle.SpacesInTab`⁽⁶⁵²⁾=0, this method does the same work as `InsertText`⁽⁵³²⁾(#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⁽²⁷³⁾.

6.1.2.2.76 TRichViewEdit.InsertText -A -W

The methods insert **text** at the position of caret.

```
procedure InsertText(const text: String;
  CaretBefore: Boolean=False);
procedure InsertTextA(const text: TRVAnsiString(993);
  CaretBefore: Boolean=False);
procedure InsertTextW(const text: TRVUnicodeString(1032);
  CaretBefore: Boolean=False);
```

Inserted text has current text⁽⁴⁷⁴⁾ and current paragraph⁽⁴⁷³⁾ style.

Text may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators⁽¹⁶²⁾, depending on value of SpacesInTab⁽⁶⁵²⁾ property of the linked⁽²⁵³⁾ RVStyle component. #12 characters add page breaks⁽²⁴⁴⁾. 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⁽²⁵³⁾.DefCodePage⁽⁶³⁵⁾.

InsertText works:

- like **InsertTextA**, in Delphi 2007 and older
- like **InsertTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus⁽¹⁵⁴⁾

See also methods:

- InsertStringTag, InsertStringATag, InsertStringWTag⁽⁵³⁰⁾.

See also properties:

- CurParaStyleNo⁽⁴⁷³⁾;
- CurTextStyleNo⁽⁴⁷⁴⁾.

See also properties of TRVStyle:

- TextStyles⁽⁶⁵⁴⁾;
- ParaStyles⁽⁶⁴⁸⁾;
- SpacesInTab⁽⁶⁵²⁾.

See also:

- Inserting items at position of caret⁽¹¹⁴⁾.

6.1.2.2.77 TRichViewEdit.InsertTextFromFile -W

The methods load text file **FileName** and insert its content at the position of caret.

InsertTextFromFile inserts text from ANSI text files, **InsertTextFromFileW** inserts text from Unicode (UTF-16) text files.

```
function InsertTextFromFile(const FileName: TRVUnicodeString1032;  
    CodePage: Cardinal=CP_ACP): Boolean;  
function InsertTextFromFileW(const FileName: TRVUnicodeString1032):  
    Boolean;
```

Inserted text has current text⁴⁷⁴ and current paragraph⁴⁷³ style.

A file may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators¹⁶², depending on value of SpacesInTab⁶⁵² property of the linked²⁵³ RVStyle component. #12 characters add page breaks²⁴⁴. 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:

- InsertOEMTextFromFile⁵²⁵;
- InsertText, -A, -W⁵³².

See also methods of TRichView:

- LoadText, LoadTextW³³⁰.

See also properties:

- CurParaStyleNo⁴⁷³;
- CurTextStyleNo⁴⁷⁴.

See also properties of TRVStyle:

- TextStyles⁶⁵⁴;
- ParaStyles⁶⁴⁸;
- SpacesInTab⁶⁵².

See also:

- Saving and loading¹²²;
- Inserting items at position of caret¹¹⁴.

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...
<i>rvcmUp</i>	up
<i>rvcmDown</i>	down
<i>rvcmLeft</i>	to the left
<i>rvcmRight</i>	to the right
<i>rvcmTop</i>	to the beginning of the document
<i>rvcmBottom</i>	to the end of the document
<i>rvcmHome</i>	to the beginning of the current line
<i>rvcmEnd</i>	to the end of the current line
<i>rvcmParaBeginning</i>	to the beginning of the current paragraph
<i>rvcmNextParaBeginning</i>	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 OnPaste⁵⁸⁴ event, allowing the programmer to insert data in custom format.

If DoDefault parameter of OnPaste was set to *False*, Paste finishes.

The method takes AcceptPasteFormats⁴⁷⁰ 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 PasteRVF⁵³⁸.

If unsuccessful, it tries to paste as RTF, see PasteRTF⁵³⁷.

If unsuccessful, it tries to paste as HTML, see PasteRTF⁵³⁶.

If unsuccessful, it tries to paste as an URL, see PasteURL⁵³⁹.

If unsuccessful, it tries to paste as text, see `PasteTextW`⁽⁵³⁸⁾.

If unsuccessful, it tries to paste image files, see `PasteGraphicFile`⁽⁵³⁵⁾ (if `rvoAssignImageFileNames` in `Options`⁽²⁴⁰⁾, file name is stored in `rvesplImageFileName` extra string property⁽¹⁰⁰⁵⁾; you can modify this file name in `OnAssignImageFileName`⁽³⁷¹⁾ event).

If unsuccessful, it tries to paste as a bitmap, see `PasteBitmap`⁽⁵³⁵⁾.

If unsuccessful, it tries to paste as a metafile picture, see `PasteMetafile`⁽⁵³⁷⁾.

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`⁽⁴⁹⁸⁾.

See also:

- `TRichView` and `Clipboard`⁽¹⁰⁸⁾.

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 image was inserted.

See also methods:

- `Paste`⁽⁵³⁴⁾;
- `InsertPicture`⁽⁵²⁶⁾.

See also properties:

- `DefaultPictureVAlign`⁽⁴⁷⁵⁾.

See also:

- `RichView` and `Clipboard`⁽¹⁰⁸⁾.

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 stored in *rvespImageFileName* extra string property⁽¹⁰⁰⁵⁾. You can modify this file name in *OnAssignImageFileName*⁽³⁷¹⁾ event.

Method type:  editing-style method for insertion.

You can use *OnDropFile*⁽⁵⁷⁶⁾ 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*⁽²⁴⁰⁾).

Supported platforms: Windows (VCL, LCL, FireMonkey)

Return value:

True, if at least one of pictures was inserted.

See also methods:

- *Paste*⁽⁵³⁴⁾;
- *InsertPicture*⁽⁵²⁶⁾.

See also properties:

- *DefaultPictureVAlign*⁽⁴⁷⁵⁾.

See also events:

- *OnAssignImageFileName*⁽³⁷¹⁾

See also:

- *RichView* and *Clipboard*⁽¹⁰⁸⁾.

6.1.2.2.82 TRichViewEdit.PasteHTML

Inserts HTML from the Clipboard at the position of the caret.

procedure *PasteHTML*;

(Introduced in version 21)

This method does nothing if the Clipboard does not contain HTML.

Parameters for loading are in *TRichView.HTMLReadProperties*⁽²³⁷⁾.

Method type:  editing-style method for insertion.

If style templates are used⁽²⁵⁶⁾, and *StyleTemplateInsertMode*⁽²⁵⁴⁾ `<>rvtIgnoreSourceStyleTemplates`, the method merges a HTML style sheet into *Style*⁽²⁵³⁾.*StyleTemplates*⁽⁶⁵²⁾, and reads text and paragraph styles according to *StyleTemplateInsertMode*⁽²⁵⁴⁾. However, most browsers copy HTML using inline CSS attributes, so *StyleTemplateInsertMode*⁽²⁵⁴⁾

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` ⁽⁴¹⁰⁾ event.

See the details about HTML style sheet in the topic about `LoadHTML` ⁽³²²⁾.

Platform notes: this method is implemented for Windows and macOS. For other platforms, it does nothing.

See also methods:

- `CanPasteHTML` ⁽⁴⁹⁹⁾;
- `InsertHTMLFromFileEd` ⁽⁵²¹⁾;
- `InsertHTMLFromStreamEd` ⁽⁵²¹⁾.

See also:

- `TRichView` and `Clipboard` ⁽¹⁰⁸⁾

6.1.2.2.83 `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 image was inserted.

See also methods:

- `Paste` ⁽⁵³⁴⁾;
- `InsertPicture` ⁽⁵²⁶⁾.

See also properties:

- `DefaultPictureVAlign` ⁽⁴⁷⁵⁾.

See also:

- `RichView` and `Clipboard` ⁽¹⁰⁸⁾.

6.1.2.2.84 `TRichViewEdit.PasteRTF`

Inserts RTF (Rich Text Format) from the Clipboard at the position of the caret.

```
procedure PasteRTF;
```

(Introduced in version 1.5)

This method does nothing if the Clipboard does not contain RTF.

If the Clipboard has RTF, and there was an error during reading, you can get additional information in `TRichView.RTFReadProperties.RTFErrorCode` ⁽⁴⁵⁷⁾ property.

Parameters for loading are in `TRichView.RTFReadProperties`⁽²⁴⁶⁾. The most important properties affecting RTF loading are `RTFReadProperties.TextStyleMode`⁽⁴⁵⁸⁾ and `ParaStyleMode`⁽⁴⁵⁶⁾. 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⁽²¹⁸⁾.

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⁽²⁵⁶⁾, and `RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=rvsAddIfNeeded`, and `StyleTemplateInsertMode`⁽²⁵⁴⁾ `<>rvstimIgnoreSourceStyleTemplates`, the method merges an RTF style sheet into `Style`⁽²⁵³⁾.`StyleTemplates`⁽⁶⁵²⁾, and reads text and paragraph styles according to `StyleTemplateInsertMode`⁽²⁵⁴⁾. The method calls `OnStyleTemplatesChange`⁽⁴¹⁰⁾ event.

See also methods:

- `CanPasteRTF`⁽⁴⁹⁹⁾;
- `InsertRTFFromFileEd`⁽⁵²⁷⁾;
- `InsertRTFFromStreamEd`⁽⁵²⁸⁾.

See also:

- `TRichView` and `Clipboard`⁽¹⁰⁸⁾

6.1.2.2.85 TRichViewEdit.PasteRVF

Inserts RVF (RichView Format) from the Clipboard at the position of caret

```
procedure PasteRVF;
```

This method does nothing if the Clipboard does not contain RVF.

Method type:  editing-style method for insertion.

See also methods:

- `CanPasteRVF`⁽⁴⁹⁹⁾;
- `Paste`⁽⁵³⁴⁾;
- `InsertRVFFromStreamEd`⁽⁵²⁹⁾.

See also:

- `TRichView` and `Clipboard`⁽¹⁰⁸⁾;
- `RVF`⁽¹²⁴⁾.

6.1.2.2.86 TRichViewEdit.PasteText -A -W

These methods Insert text from the Clipboard at the position of caret

```
procedure PasteText;
```

```
procedure PasteTextA;
```

```
procedure PasteTextW;
```

PasteTextA pastes text in CF_TEXT format. **PasteText** and **PasteTextW** paste text in CF_UNICODETEXT.

In modern versions of Windows, Windows automatically converts Unicode and ANSI text in the Clipboard to each other, using the language of the keyboard layout that was used at the moment of copying, so using **PasteTextA** does not make sense.

These methods do nothing if the Clipboard does not contain text.

Text is inserted at the position of caret, using the current text⁽⁴⁷⁴⁾ and paragraph⁽⁴⁷³⁾ styles.

Text from the Clipboard may contain special characters: CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators⁽¹⁶²⁾, depending on value of SpacesInTab⁽⁶⁵²⁾ property of the linked⁽²⁵³⁾ RVStyle component. #12 characters add page breaks⁽²⁴⁴⁾. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported. **PasteTextW** supports the Unicode byte order marks characters (if it's present, it must be the first character).

Methods type:  editing-style method for insertion.

 **Unicode note:**

- **PasteTextA:** itext from the Clipboard will be converted to Unicode (conversion is based on the charset of the current text style).

See also methods:

- Paste⁽⁵³⁴⁾;
- InsertText, -A, -W⁽⁵³²⁾.

See also properties:

- CurParaStyleNo⁽⁴⁷³⁾;
- CurTextStyleNo⁽⁴⁷⁴⁾.

See also properties of TRVStyle:

- TextStyles⁽⁶⁵⁴⁾;
- ParaStyles⁽⁶⁴⁸⁾;
- SpacesInTab⁽⁶⁵²⁾.

See also:

- RichView and Clipboard⁽¹⁰⁸⁾.

6.1.2.2.87 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⁽³⁸⁸⁾ 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⁽⁹¹⁾).

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⁽⁵³⁴⁾.

See also:

- TRichView and Clipboard⁽¹⁰⁸⁾.

6.1.2.2.88 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⁽⁵⁴⁰⁾ and RedoName⁽⁵⁴⁰⁾.

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: **I** editing-style.

See also:

- Undo⁽⁵⁶⁷⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.2.89 TRichViewEdit.RedoAction

Returns type of the operation to be redone⁽⁵⁴⁰⁾ next.

function RedoAction: TRVUndoType⁽¹⁰³¹⁾;

(Introduced in version 1.3)

See TRVUndoType⁽¹⁰³¹⁾ for the list of operations. If the returned value is *rvutCustom*, you can use RedoName⁽⁵⁴⁰⁾.

See also:

- UndoAction⁽⁵⁶⁷⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.2.90 TRichViewEdit.RedoName

Returns name of the operation to be redone⁽⁵⁴⁰⁾ next.

function RedoName: TRVUnicodeString⁽¹⁰³²⁾;

(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⁽⁵⁴⁰⁾).

Returns non-empty string only for custom redo action, initiated by BeginUndoCustomGroup⁽⁴⁹⁶⁾.

See also:

- UndoName⁽⁵⁶⁸⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.2.91 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⁽⁵⁰³⁾.

This method is used in conjunction with InsertCheckpoint⁽⁵¹⁶⁾ and GetCheckpointAtCaret⁽⁵⁰³⁾. The alternative set of methods is: SetCurrentCheckpointInfo⁽⁵⁵³⁾, GetCurrentCheckpoint⁽⁵⁰⁶⁾ and RemoveCurrentCheckpoint⁽⁵⁴¹⁾.

Method type: **I** editing-style, unlike RemoveCheckpoint⁽³³²⁾.

See also:

- "Checkpoints"⁽⁸⁷⁾.

6.1.2.2.92 TRichViewEdit.RemoveCheckpointEd

Editing-style method for deleting *checkpoint* associated with the **ItemNo**-th item.

procedure RemoveCheckpointEd(ItemNo: Integer);

Method type: **I** editing-style, unlike RemoveCheckpoint⁽³³²⁾.

See also methods of TRichViewEdit:

- RemoveCurrentCheckpoint⁽⁵⁴¹⁾;
- SetCheckpointInfoEd⁽⁵⁴⁹⁾.

See also methods of TRichView:

- GetItemCheckpoint⁽²⁹⁸⁾;
- RemoveCheckpoint⁽³³²⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- "Checkpoints"⁽⁸⁷⁾.

6.1.2.2.93 TRichViewEdit.RemoveCurrentCheckpoint

Deletes *checkpoint* associated with the item at the position of caret.

procedure RemoveCurrentCheckpoint ;

RemoveCurrentCheckpoint is equivalent to TopLevelEditor⁽⁴⁸¹⁾.RemoveCheckpointEd⁽⁵⁴¹⁾ (TopLevelEditor⁽⁴⁸¹⁾.CurItemNo⁽⁴⁷²⁾).

Method type: **I** editing-style.

See also methods:

- GetCurrentCheckpoint⁽⁵⁰⁶⁾;
- SetCurrentCheckpointInfo⁽⁵⁵³⁾;
- RemoveCheckpointEd⁽⁵⁴¹⁾.

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `"Checkpoints"`⁽⁸⁷⁾.

6.1.2.2.94 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`⁽²⁴⁴⁾), if this flag was set.

Method type:  editing-style.

See also:

- `InsertPageBreak`⁽⁵²⁵⁾.

See also properties of RichView:

- `PageBreaksBeforeItems`⁽²⁴⁴⁾;
- `Options`⁽²⁴⁰⁾ (`rvoShowPageBreaks`).

See also properties of RVStyle:

- `PageBreakColor`⁽⁶⁴⁷⁾.

6.1.2.2.95 TRichViewEdit.RemoveLists

Removes markers⁽¹⁷⁴⁾ (bullets and numbering) from the selected paragraphs.

procedure `RemoveLists`(`AResursive`: Boolean);

(introduced in version 1.7)

Parameter:

AResursive – reserved for future use. Set it to *False*.

Method type:  editing-style.

See also:

- `ApplyListStyle`⁽⁴⁹⁰⁾;
- `ChangeListLevels`⁽⁵⁰⁰⁾.

See also methods of TCustomRichView:

- `SetListMarkerInfo`⁽³⁶²⁾;
- `GetListMarkerInfo`⁽³⁰⁷⁾.

6.1.2.2.96 TRichViewEdit.ResizeControl

Resizes control⁽¹⁶⁸⁾ in the **ItemNo**-th item, and quickly reformats the affected part of the document.

procedure `ResizeControl`(`ItemNo`, `NewWidth`, `NewHeight`: TRVCoord⁽⁹⁹⁸⁾);

The **ItemNo**-th item must be of "inserted control"⁽¹⁶⁸⁾ type (*rvsComponent*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception.

The control's width will be set to **NewWidth**, height to **NewHeight**.

Method type:  editing-style.

See also methods:

- *ResizeCurrentControl*⁽⁵⁴³⁾;
- *SetControlInfoEd*⁽⁵⁵⁰⁾;
- *AdjustControlPlacement*⁽⁴⁸⁹⁾;
- *GetItemStyle*⁽³⁰²⁾.

See also properties:

- *ItemCount*⁽²³⁷⁾.

See also:

- *Modifying RichView items*⁽¹¹²⁾;
- *Item types*⁽⁸⁵⁾.

6.1.2.2.97 TRichViewEdit.ResizeCurrentControl

Resizes control⁽¹⁶⁸⁾ at the position of caret and quickly reformats the affected part of the document.

procedure *ResizeCurrentControl*(*NewWidth*, *NewHeight*: *TRVCoord*⁽⁹⁹⁸⁾);

(*introduced in version 1.4*)

ResizeCurrentControl(...) is equivalent to *TopLevelEditor*⁽⁴⁸¹⁾.*ResizeControl*⁽⁵⁴²⁾(*TopLevelEditor*⁽⁴⁸¹⁾.*CurlItemNo*⁽⁴⁷²⁾, ...).

Method type:  editing-style.

See also methods:

- *ResizeControl*⁽⁵⁴²⁾;
- *SetControlInfoEd*⁽⁵⁵⁰⁾;
- *AdjustControlPlacement*⁽⁴⁸⁹⁾;
- *GetItemStyle*⁽³⁰²⁾.

See also properties:

- *ItemCount*⁽²³⁷⁾.

See also:

- *Modifying RichView items*⁽¹¹²⁾;
- *Item types*⁽⁸⁵⁾.

6.1.2.2.98 TRichViewEdit.SearchText -W

The methods search for the substring **s** in the document.

function *SearchText*(*s*: *String*;
SrchOptions: *TRVSearchOptions*⁽⁹⁹⁹⁾): *Boolean*;

function *SearchTextA*(*s*: *TRVAnsiString*⁽⁹⁹³⁾;
SrchOptions: *TRVSearchOptions*⁽⁹⁹⁹⁾): *Boolean*;

```
function SearchTextW(s: TRVUnicodeString1032;
  SrchOptions: TRVESearchOptions999): Boolean;
```

When found, these methods select substring and move the caret to the end of selection (this behavior can be modified in OnTextFound⁴¹⁰ 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 position³⁶⁷ if used instead of selection or a caret position.

If *rvseoMultiItem* is included, the search can match substrings of several text items¹⁶¹. If not included, the text is searched in each text item separately. For example, if the document contains the text **Hello**, 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 (Style²⁵³.DefCodePage⁶³⁵) and calls **SearchTextW**.

SearchText works:

- like **SearchTextA**, in Delphi 2007 and older
- like **SearchTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus¹⁵⁴

These methods can be used for search-and-replace operation as well. **SearchText** selects the found string, InsertText⁵³² replaces it with the new one. See important note about search and replace in down-up direction in the topic about InsertText⁵³².

*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, OnTextFound⁴¹⁰ 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 OnTextFound⁴¹⁰ 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⁽³⁴⁶⁾;
- GetRVESearchOptions⁽⁹⁸⁶⁾ global function;
- Searching in TRichView and TRichViewEdit⁽¹¹⁰⁾.

6.1.2.2.99 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⁽⁴⁰⁸⁾ event.

This method must be called only when the document is formatted.

See also methods:

- SelectCurrentWord⁽⁵⁴⁵⁾;
- GetSelText⁽³¹³⁾.

See also:

- Selecting in RichView document⁽¹⁰⁷⁾.

6.1.2.2.100 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⁽⁴⁰⁸⁾ event.

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters⁽²²⁸⁾ property.

This method must be called only when the document is formatted.

See also methods:

- SelectWordAt⁽³⁴⁹⁾;
- GetSelText⁽³¹³⁾;
- SelectCurrentLine⁽⁵⁴⁵⁾.

See also:

- Selecting in RichView document⁽¹⁰⁷⁾.

6.1.2.2.101 TRichViewEdit.Set*PropertyEd

Editing-style methods for changing properties of the editor.

```
procedure SetIntPropertyEd(Prop: TRVIntProperty(1013);  
  Value: Integer; AutoReformat: Boolean);
```

```
procedure SetStrPropertyEd(Prop: TRVStrProperty(1026);  
  const Value: TRVUnicodeString(1032); AutoReformat: Boolean);
```

```
procedure SetFloatPropertyEd(Prop: TRVFloatProperty(1007);
```

```
Value: TRVLength(1015); 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⁽⁴⁹⁹⁾ 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(497)(rvutLayout);
  MyRichViewEdit.SetUndoGroupMode(567)(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(567)(False);
  MyRichViewEdit.Reformat(332);
  MyRichViewEdit.Change(499);
end;
```

See also methods:

- SetBackgroundImageEd⁽⁵⁴⁶⁾.

6.1.2.2.102 TRichViewEdit.SetBackgroundImageEd

An editing-style method for changing BackgroundBitmap⁽²²²⁾.

```
procedure SetBackgroundImageEd(Graphic: TRVGraphic(970); AutoReformat: Boolean);
```

(introduced in version 15)

Parameters:

Graphic – a graphic to assign to BackgroundBitmap⁽²²²⁾. 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⁽⁴⁹⁹⁾ is called. Normally, this method is called with **AutoReformat=True** if you change a single property. If you call this method together with other methods in a group, this parameter must be *False*, and you must call Change yourself.

Method type:  editing-style (unlike direct assignment to BackgroundBitmap⁽²²²⁾).

Example:

```
if MyRichViewEdit.CanChange then
begin
  MyRichViewEdit.BeginUndoGroup(497)(rvutBackground);
  MyRichViewEdit.SetUndoGroupMode(567)(True);
  MyRichViewEdit.SetIntPropertyEd(545)(rvipBackgroundStyle, Ord(bsTiled),
    False);
  MyRichViewEdit.SetBackgroundImageEd(Imagel.Picture.Graphic, False);
  MyRichViewEdit.SetUndoGroupMode(567)(False);
  MyRichViewEdit.Invalidate;
  MyRichViewEdit.Change(499);
end;
```

See also methods:

- SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd⁽⁵⁴⁵⁾.

6.1.2.2.103 TRichViewEdit.SetBreakInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a *break*⁽¹⁶⁷⁾.

```
procedure SetBreakInfoEd(ItemNo: Integer; AWidth: TRVStyleLength(1027);
  AStyle: TRVBreakStyle(995); AColor: TRVColor(996); const ATag: TRVTag(1029));
```

Parameters:

ItemNo – index of the item. The item must be of *break*⁽¹⁶⁷⁾ type (*rvsBreak*⁽¹⁰⁵⁹⁾), otherwise the method raises ERichViewError⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽³⁶⁵⁾), and call SetBreakInfoEd of this inplace-editor. Alternatively, you can use SetCurrentBreakInfo⁽⁵⁵¹⁾.

AWidth – line width (or rectangle height). This value is measured in Style⁽²⁵³⁾.Units⁽⁶⁵⁵⁾.

AStyle – visual style of this *break*, see TRVBreakStyle⁽⁹⁹⁵⁾ for possible values.

AColor – line color. If it is equal to *rvclNone*⁽¹⁰³⁸⁾, Style⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾[0].Color⁽⁷⁰¹⁾ is used.

ATag – *tag* of the item. You can use value returned by GetBreakInfo⁽²⁸⁹⁾ or GetItemTag⁽³⁰²⁾ for this item.

Instead of this method, you can use SetItemExtraIntPropertyExEd⁽⁵⁶²⁾ and SetItemTagEd⁽⁵⁶³⁾ methods.

Method type: **I** editing-style (unlike `SetBreakInfo`⁽³⁵¹⁾).

Additional item properties are assigned by the methods `SetItemExtraIntPropertyEd`⁽⁵⁶²⁾ and `SetItemExtraStrPropertyEd`⁽⁵⁶²⁾.

When possible, use `SetCurrentBreakInfo`⁽⁵⁵¹⁾ instead of this method.

See also methods of TCustomRichView:

- `GetBreakInfo`⁽²⁸⁹⁾;
- `GetItemStyle`⁽³⁰²⁾;
- `SetBreakInfo`⁽³⁵¹⁾.

See also methods:

- `SetCurrentBreakInfo`⁽⁵⁵¹⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also:

- [Modifying RichView items](#)⁽¹¹²⁾;
- [Item types](#)⁽⁸⁵⁾;
- ["Tags"](#)⁽⁹¹⁾.

6.1.2.2.104 TRichViewEdit.SetBulletInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of TRichView, if this item is a *bullet*⁽¹⁷⁰⁾ or *hotspot*⁽¹⁷²⁾.

VCL and LCL:

```
procedure SetBulletInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString(1032); AImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag(1029));
```

FireMonkey:

```
procedure SetBulletInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString(1032); AImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag(1029);
  ImageWidth, ImageHeight: TRVStyleLength(1027));
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *bullet*⁽¹⁷⁰⁾ or *hotspot*⁽¹⁷²⁾ type (*rvsBullet* or *rvsHotspot*⁽¹⁰⁵⁹⁾), otherwise the method raises `ERichViewError`⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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`⁽³⁶⁵⁾), and call `SetBulletInfoEd` of this inplace-editor. Alternatively, you can use `SetCurrentBulletInfo`⁽⁵⁵²⁾.

AName – name of *bullet*⁽¹⁷⁰⁾, any string without line break (CR, LF) characters. It can also be set using `SetItemTextEdW`⁽⁵⁶⁴⁾ method.

AlmageList – not used, reserved, set it to *nil*.

AlmageIndex – index of image in image list. It can also be set using `SetItemExtraIntPropertyExEd`⁽⁵⁶²⁾ method.

ATag – *tag* of the item. You can use value returned by `GetBulletInfo`⁽²⁸⁹⁾ or `GetItemTag`⁽³⁰²⁾ for this item. The *tag* can also be set by `SetItemTagEd`⁽⁵⁶³⁾ 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: **I** editing-style (unlike `SetBulletInfo`⁽³⁵²⁾).

Additional item properties are assigned by the methods `SetItemExtraIntPropertyEd`⁽⁵⁶²⁾ and `SetItemExtraStrPropertyEd`⁽⁵⁶²⁾.

When possible, use `SetCurrentBulletInfo`⁽⁵⁵²⁾ instead of this method.

See also methods of TCustomRichView:

- `GetBulletInfo`⁽²⁸⁹⁾;
- `GetItemStyle`⁽³⁰²⁾;
- `SetBulletInfo`⁽³⁵²⁾.

See also methods:

- `SetCurrentBulletInfo`⁽⁵⁵²⁾.

See also properties:

- `ItemCount`⁽²³⁷⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `"Tags"`⁽⁹¹⁾.

6.1.2.2.105 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(1029);
const AName: TRVUnicodeString(1032); ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to `ItemCount`⁽²³⁷⁾-1. Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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 (`SelfSelectionBounds`⁽³⁶⁵⁾), and call `SetCheckpointInfoEd` of this inplace-editor. Alternatively, you can use `SetCurrentCheckpointInfo`⁽⁵⁵³⁾.

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⁽³⁷²⁾ event for this *checkpoint*.

Method type: **I** editing-style (unlike SetCheckpointInfo⁽³⁵³⁾).

When possible, use SetCurrentCheckpointInfo⁽⁵⁵³⁾ instead of this method.

See also methods of TCustomRichView:

- GetItemCheckpoint⁽²⁹⁸⁾;
- SetCheckpointInfo⁽³⁵³⁾.

See also methods:

- SetCurrentCheckpointInfo⁽⁵⁵³⁾.

See also properties:

- ItemCount⁽²³⁷⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- "Checkpoints"⁽⁸⁷⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.106 TRichViewEdit.SetControllInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is "inserted control"⁽¹⁶⁸⁾.

```
procedure SetControllInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString(1032); AAlign: TRVAlign(1033);
  const ATag: TRVTag(1029));
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control⁽¹⁶⁸⁾ type (*rvcComponent*⁽¹⁰⁵⁹⁾), otherwise the method raises ERichViewError⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to ItemCount⁽²³⁷⁾-1, GetItemStyle⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽³⁶⁵⁾), and call SetControllInfoEd of this inplace-editor. Alternatively, you can use SetCurrentControllInfo⁽⁵⁵⁴⁾.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW⁽⁵⁶⁴⁾ method. This is not a Name property of the control!

AAlign – vertical alignment of this item, see TRVAlign⁽¹⁰³³⁾ for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetControllInfo⁽²⁹³⁾ or GetItemTag⁽³⁰²⁾ for this item. The *tag* can also be set by SetItemTagEd⁽⁵⁶³⁾ method.

Method type: **I** editing-style (unlike `SetControlInfo`³⁵⁴).

Additional item properties are assigned by the methods `SetItemExtraIntPropertyEd`⁵⁶² and `SetItemExtraStrPropertyEd`⁵⁶².

When possible, use `SetCurrentControlInfo`⁵⁵⁴ instead of this method.

See also methods of `TCustomRichView`:

- `GetControlInfo`²⁹³;
- `GetItemStyle`³⁰²;
- `SetControlInfo`³⁵⁴.

See also methods:

- `SetCurrentControlInfo`⁵⁵⁴;
- `ResizeControl`⁵⁴²;
- `AdjustControlPlacement`⁴⁸⁹.

See also properties:

- `ItemCount`²³⁷.

See also:

- `Modifying RichView items`¹¹²;
- `Item types`⁸⁵;
- `"Tags"`⁹¹.

6.1.2.2.107 `TRichViewEdit.SetCurrentBreakInfo`

Changes properties for the item at the position of caret, if this item is a *break*¹⁶⁷

```
procedure SetCurrentBreakInfo(AWidth: TRVStyleLength1027;
  AStyle: TRVBreakStyle995; AColor: TRVColor996; const ATag: TRVTag1029);
```

`SetCurrentBreakInfo(...)` is equivalent to `TopLevelEditor`⁴⁸¹.`SetBreakInfoEd`⁵⁴⁷ (`TopLevelEditor`⁴⁸¹.`CurItemNo`⁴⁷², ...).

This method can be used if `CurItemStyle`⁴⁷³ returns *rvsBreak*¹⁰⁵⁹.

Parameters:

AWidth – line width (or rectangle height). This value is measured in `Style`²⁵³.`Units`⁶⁵⁵.

AStyle – visual style of this *break*, see `TRVBreakStyle`⁹⁹⁵ for possible values.

AColor – line color. If it is equal to *rvclNone*¹⁰³⁸, `Style`²⁵³.`TextStyles`⁶⁵⁴[0].`Color`⁷⁰¹ is used.

ATag – *tag* of the item. You can use value returned by `GetCurrentBreakInfo`⁵⁰⁴ or `GetCurrentTag`⁵¹⁴ for this item.

Instead of this method, you can use `SetCurrentItemExtraIntPropertyEx`⁵⁵⁶ and `SetCurrentTag`⁵⁶⁰ methods.

Method type: **I** editing-style.

Additional item properties are assigned by the methods `SetCurrentItemExtraIntProperty`⁵⁵⁶ and `SetCurrentItemExtraStrProperty`⁵⁵⁷.

See also methods:

- GetCurrentBreakInfo⁽⁵⁰⁴⁾;
- SetBreakInfoEd⁽⁵⁴⁷⁾.

See also properties:

- CurlItemNo⁽⁴⁷²⁾;
- CurlItemStyle⁽⁴⁷³⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.108 TRichViewEdit.SetCurrentBulletInfo

Changes properties for the item at the position of caret, if this item is a *bullet*⁽¹⁷⁰⁾ or *hotspot*⁽¹⁷²⁾.

VCL and LCL:

```
procedure SetCurrentBulletInfo(const AName: TRVUnicodeString(1032);
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag(1029));
```

(changed in version 18)

FireMonkey:

```
procedure SetCurrentBulletInfo(const AName: TRVUnicodeString(1032);
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag(1029);
  AImageWidth, AImageHeight: TRVStyleLength(1027));
```

SetCurrentBulletInfo(...) is equivalent to `TopLevelEditor(481).SetBulletInfoEd(548)(TopLevelEditor(481).CurlItemNo(472), ...)`.

This method can be used if `CurlItemStyle(473)` returns *rvsBullet* or *rvsHotspot*⁽¹⁰⁵⁹⁾.

Parameters:

AName – name of *bullet*⁽¹⁷⁰⁾, any string without line break (CR, LF) characters. It can also be set using `SetCurrentItemTextW(557)` method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using `SetCurrentItemExtraIntPropertyEx(556)` method.

ATag – *tag* of the item. You can use value returned by `GetCurrentBulletInfo(505)` or `GetCurrentTag(514)` for this item. The *tag* can also be set by `SetCurrentTag(560)` 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`⁵⁵⁶ and `SetCurrentItemExtraStrProperty`⁵⁵⁷.

See also methods:

- `GetCurrentBulletInfo`⁵⁰⁵;
- `SetBulletInfoEd`⁵⁴⁸.

See also properties:

- `CurlItemNo`⁴⁷²;
- `CurlItemStyle`⁴⁷³.

See also:

- `Modifying RichView items`¹¹²;
- `Item types`⁸⁵;
- `"Tags"`⁹¹.

6.1.2.2.109 TRichViewEdit.SetCurrentCheckpointInfo

Creates or modifies *checkpoint* associated with the item at the position of caret.

```
function SetCurrentCheckpointInfo(const ATag: TRVTag1029;
const AName: TRVUnicodeString1032; ARaiseEvent: Boolean): Boolean;
```

(changed in version 18 and 19)

`SetCurrentCheckpointInfo(...)` is equivalent to `TopLevelEditor`⁴⁸¹.`SetCheckpointInfoEd`⁵⁴⁹ (`TopLevelEditor`⁴⁸¹.`CurlItemNo`⁴⁷², ...).

If the item at the position of caret already has a *checkpoint* (see `GetCurrentCheckpoint`⁵⁰⁶), its properties are modified; otherwise, a new *checkpoint* is created.

This method is used in conjunction with `GetCurrentCheckpoint`⁵⁰⁶ and `RemoveCurrentCheckpoint`⁵⁴¹. The alternative set of methods is: `InsertCheckpoint`⁵¹⁶, `GetCheckpointAtCaret`⁵⁰³ and `RemoveCheckpointAtCaret`⁵⁴¹.

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`³⁷² 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`⁵⁰⁶, `RemoveCurrentCheckpoint`⁵⁴¹;

- InsertCheckpoint⁽⁵¹⁶⁾, GetCheckpointAtCaret⁽⁵⁰³⁾, RemoveCheckpointAtCaret⁽⁵⁴¹⁾;
- SetCheckpointInfoEd⁽⁵⁴⁹⁾.

See also properties:

- CurlItemNo⁽⁴⁷²⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- "Checkpoints"⁽⁸⁷⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.110 TRichViewEdit.SetCurrentControllInfo

Modifies the item at the position of caret, if this item is an inserted control⁽¹⁶⁸⁾.

```
procedure SetCurrentControllInfo(const AName: TRVUnicodeString(1032);
  AAlign: TRVAlign(1033); const ATag: TRVTag(1029));
```

(changed in version 18)

SetCurrentControllInfo(...) is equivalent to TopLevelEditor⁽⁴⁸¹⁾.**SetControllInfoEd**⁽⁵⁵⁰⁾
(TopLevelEditor⁽⁴⁸¹⁾.CurlItemNo⁽⁴⁷²⁾, ...).

This method can be used if CurlItemStyle⁽⁴⁷³⁾ returns *rvsComponent*⁽¹⁰⁵⁹⁾.

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemText⁽⁵⁵⁷⁾ method. This is not a Name property of the control!

AAlign – vertical alignment of this item, see TRVAlign⁽¹⁰³³⁾ for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetCurrentControllInfo⁽⁵⁰⁶⁾ or GetCurrentTag⁽⁵¹⁴⁾ for this item. The *tag* can also be set by SetCurrentTag⁽⁵⁶⁰⁾ method.

Method type:  editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty⁽⁵⁵⁶⁾ and SetCurrentItemExtraStrProperty⁽⁵⁵⁷⁾.

See also methods:

- GetCurrentControllInfo⁽⁵⁰⁶⁾;
- SetControllInfoEd⁽⁵⁵⁰⁾.

See also properties:

- CurlItemNo⁽⁴⁷²⁾.
- CurlItemStyle⁽⁴⁷³⁾.

See also:

- Modifying RichView items⁽¹¹²⁾;
- Item types⁽⁸⁵⁾;
- "Tags"⁽⁹¹⁾.

6.1.2.2.111 TRichViewEdit.SetCurrentHotspotInfo

Modifies the item at the position of caret, if this item is a *hotspot*⁽¹⁷²⁾.

VCL and LCL:

```
procedure SetCurrentHotspotInfo(const AName: TRVUnicodeString(1032);  
    AImageIndex, AHotImageIndex: Integer; AImageList: TCustomImageList;  
    const ATag: TRVTag(1029));
```

FireMonkey:

```
procedure SetCurrentHotspotInfo(const AName: TRVUnicodeString(1032);  
    AImageIndex, AHotImageIndex: Integer; AImageList: TCustomImageList;  
    const ATag: TRVTag(1029);  
    AImageWidth, AImageHeight: TRVStyleLength(1027));
```

(changed in version 18)

SetCurrentHotspotInfo(...) is equivalent to `TopLevelEditor(481).SetHotspotInfoEd(560)` (`TopLevelEditor(481).CurlItemNo(472), ...`).

This method can be used if `CurlItemStyle(473)` returns `rvsHotspot(1059)`.

Parameters:

AName – name of *hotspot*⁽¹⁷²⁾, any string without line break (CR, LF) characters. It can also be set using `SetCurrentItemTextW(557)` method.

AImageList – not used, reserved, set it to *nil*..

AImageIndex – index of image in the image list. It can also be set using `SetCurrentItemExtraIntPropertyEx(556)` 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(556)` method.

ATag – *tag* of the item. You can use value returned by `GetCurrentHotspotInfo(507)` or `GetCurrentTag(514)` for this item. The *tag* can also be set by `SetCurrentTag(560)` 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(556)` and `SetCurrentItemExtraStrProperty(557)`.

See also methods:

- `GetCurrentHotspotInfo(507)`;
- `SetHotspotInfoEd(560)`.

See also properties:

- `CurlItemNo(472)`;

- `CurItemStyle`⁽⁴⁷³⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `"Tags"`⁽⁹¹⁾.

6.1.2.2.112 TRichViewEdit.SetCurrentItemExtraIntProperty -Ex

The methods change value of an integer property for the item at the position of caret.

```
procedure SetCurrentItemExtraIntProperty(Prop: TRVExtraItemProperty(1000);
  Value: Integer; AutoReformat: Boolean);
```

```
procedure SetCurrentItemExtraIntPropertyEx(Prop, Value: Integer;
  AutoReformat: Boolean);
```

(introduced in versions 1.7 and 15)

`SetCurrentItemExtraIntProperty[Ex](...)` is equivalent to `TopLevelEditor`⁽⁴⁸¹⁾.
`.SetItemExtraIntProperty[Ex]Ed`⁽⁵⁶²⁾ (`TopLevelEditor`⁽⁴⁸¹⁾.`CurItemNo`⁽⁴⁷²⁾, ...).

These methods set a new **Value** of the item's property identified by **Prop**.

In `SetCurrentItemExtraIntProperty`, **Prop**'s type is `TRVExtraItemProperty`⁽¹⁰⁰⁰⁾. 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⁽¹⁰⁵¹⁾

If `AutoReformat=True`, these methods automatically reformat the document and call `OnChange`⁽⁵⁷²⁾.

If `AutoReformat=False` (this can be useful if you set several properties of an item at once), call `Change`⁽⁴⁹⁹⁾ 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`⁽⁴⁹⁵⁾ and `EndItemModify`⁽⁵⁰³⁾);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call `Reformat`⁽³³²⁾).

Method type:  editing-style.

See also methods:

- `GetCurrentItemExtraIntProperty[Ex]`⁽⁵¹⁰⁾;
- `SetItemExtraIntProperty[Ex]Ed`⁽⁵⁶²⁾;
- `SetCurrentItemExtraStrProperty`⁽⁵⁵⁷⁾;

See also methods of TCustomRichView:

- `SetItemExtraIntProperty[Ex]`⁽³⁵⁸⁾.

6.1.2.2.113 TRichViewEdit.SetCurrentItemExtraStrProperty -Ex

The methods change a value of a string property for the item at the position of caret

```
procedure SetCurrentItemExtraStrProperty(
  Prop: TRVExtraItemStrProperty1005;
  const Value: TRVUnicodeString1032; AutoReformat: Boolean);
procedure SetCurrentItemExtraStrPropertyEx(Prop: Integer;
  const Value: TRVUnicodeString1032; AutoReformat: Boolean);
```

(introduced in versions 1.9 and 15; changed in version 18)

SetCurrentItemExtraStrProperty[Ex](...) is equivalent to `TopLevelEditor481.SetItemExtraStrProperty[Ex]Ed562(TopLevelEditor481.CurlItemNo472, ...)`.

This method sets a new **Value** of the item's property identified by **Prop**.

In **SetCurrentItemExtraStrProperty**, **Prop**'s type is `TRVExtraItemStrProperty1005`. 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*** constants1056`

If **AutoReformat=True**, these methods automatically reformat the document and call `OnChange572`.

If **AutoReformat=False** (this can be useful if you set several properties of an item at once), call `Change499` 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 `BeginItemModify495` and `EndItemModify503`);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call `Reformat332`).

Method type:  editing-style.

See also methods:

- `GetCurrentItemExtraStrProperty[Ex]510`;
- `SetItemExtraStrProperty[Ex]Ed562`;
- `SetCurrentItemExtraIntProperty[Ex]556`.

See also methods of TCustomRichView:

- `SetItemExtraStrProperty[Ex]359`.

6.1.2.2.114 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: TRVAnsiString993);
procedure SetCurrentItemTextW(const s: TRVUnicodeString1032);
```

(Introduced in version 1.4, 1.7)

- **SetCurrentItemText(s)** is equivalent to `TopLevelEditor481.SetItemTextEd564(TopLevelEditor481.CurlItemNo472, s)`.

- **SetCurrentItemTextA**(s) is equivalent to `TopLevelEditor481.SetItemTextEdA564` (`TopLevelEditor481.CurlItemNo472, s`).
- **SetCurrentItemTextW**(s) is equivalent to `TopLevelEditor481.SetItemTextEdW564` (`TopLevelEditor481.CurlItemNo472, 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 Charset⁷⁰⁰. If it is equal to DEFAULT_CHARSET, and for non-text items, `Style253.DefCodePage635` is used.

SetCurrentItemText works:

- like **SetCurrentItemTextA**, in Delphi 2007 and older
- like **SetCurrentItemTextW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus¹⁵⁴

Methods type:  editing-style.

See also methods:

- `GetCurrentItemText -A -W511`;
- `SetItemTextEd -A -W564`.

See also methods of TCustomRichView:

- `SetItemText -A -W360`.

See also properties:

- `CurlItemNo472`;
- `CurlItemStyle473`.

See also:

- Obtaining RichView items¹¹¹;
- Item types⁸⁵;
- "Tags"⁹¹;
- Unicode in RichView¹³⁰.

6.1.2.2.115 TRichViewEdit.SetCurrentItemVAlign

Changes vertical alignment (position relative to the line) of the item at the position of caret.

procedure `SetCurrentItemVAlign(VAlign: TRVVAlign1033);`

(introduced in version 12)

SetCurrentItemVAlign(VAlign) is equivalent to `TopLevelEditor481.SetItemVAlignEd565` (`TopLevelEditor481.CurlItemNo472, VAlign`).

Method type:  editing-style.

This method must not be called if the current item is one of: text items⁽¹⁶¹⁾, tables⁽¹⁷³⁾, *breaks*⁽¹⁶⁷⁾, tabs⁽¹⁶²⁾, list markers⁽¹⁷⁴⁾, footnotes⁽¹⁸⁰⁾ and endnotes⁽¹⁷⁸⁾.

See also methods:

- `GetCurrentItemVAlign`⁽⁵¹²⁾;
- `SetItemVAlignEd`⁽⁵⁶⁵⁾.

See also properties:

- `CurlItemNo`⁽⁴⁷²⁾.

See also:

- Modifying RichView items⁽¹¹²⁾.

6.1.2.2.116 TRichViewEdit.SetCurrentPictureInfo

Modifies the item at the position of caret, if this item is a picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾.

```
procedure SetCurrentPictureInfo(const AName: TRVUnicodeString(1032);
  Agr: TRVGraphic(970); AAlign: TRVVAlign(1033); const ATag: TRVTag(1029));
```

(changed in version 18)

SetCurrentPictureInfo(...) is equivalent to `TopLevelEditor`⁽⁴⁸¹⁾.**SetPictureInfoEd**⁽⁵⁶⁵⁾ (`TopLevelEditor`⁽⁴⁸¹⁾.`CurlItemNo`⁽⁴⁷²⁾, ...).

This method can be used if `CurlItemStyle`⁽⁴⁷³⁾ returns *rvsPicture*⁽¹⁰⁵⁹⁾.

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using `SetCurrentItemTextW`⁽⁵⁵⁷⁾ 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`⁽⁵¹³⁾ (the image is shared if *rvspShared* extra item property⁽¹⁰⁰⁰⁾ is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by `GetCurrentPictureInfo`⁽⁵¹³⁾, the previously used graphic object is freed.

AAlign – vertical alignment of the picture.

ATag – *tag* of the item. You can use value returned by `GetCurrentPictureInfo`⁽⁵¹³⁾ or `GetCurrentTag`⁽⁵¹⁴⁾ for this item. The *tag* can also be set by `SetCurrentTag`⁽⁵⁶⁰⁾ method.

Method type:  editing-style.

Additional item properties are assigned by the methods `SetCurrentItemExtraIntProperty`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrProperty`⁽⁵⁵⁷⁾.

See also methods:

- `GetCurrentPictureInfo`⁽⁵¹³⁾;
- `SetPictureInfoEd`⁽⁵⁶⁵⁾;
- `SetCurrentItemExtraIntProperty`⁽⁵⁵⁶⁾.

See also properties:

- `CurItemNo`⁽⁴⁷²⁾;
- `CurItemStyle`⁽⁴⁷³⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `Item types`⁽⁸⁵⁾;
- `"Tags"`⁽⁹¹⁾.

6.1.2.2.117 TRichViewEdit.SetCurrentTag

Changes tag of the item at the position of caret.

procedure `SetCurrentTag(ATag: TRVTag(1029))`;

`SetCurrentTag(Tag)` is equivalent to `TopLevelEditor(481).SetItemTagEd(563)(TopLevelEditor(481).CurItemNo(472), Tag)`.

Method type:  editing-style.

See also methods:

- `GetCurrentTag`⁽⁵¹⁴⁾;
- `SetItemTagEd`⁽⁵⁶³⁾.

See also properties:

- `CurItemNo`⁽⁴⁷²⁾.

See also:

- `Modifying RichView items`⁽¹¹²⁾;
- `"Tags"`⁽⁹¹⁾.

6.1.2.2.118 TRichViewEdit.SetHotspotInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a *hotspot*⁽¹⁷²⁾.

VCL and LCL:

```
procedure SetHotspotInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString(1032);
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag(1029));
```

(changed in version 18)

FireMonkey:

```
procedure SetHotspotInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString(1032);
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag(1029);
  ImageWidth, ImageHeight: TRVStyleLength(1027));
```

Parameters:

ItemNo – index of the item. The item must be of *hotspot*⁽¹⁷²⁾ type (*rvsHotspot*⁽¹⁰⁵⁹⁾), otherwise the method raises `ERichViewError`⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to `ItemCount`⁽²³⁷⁾-1, `GetItemStyle`⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells ⁸⁹⁵) 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 ³⁸⁵), and call SetHotspotInfoEd of this inplace-editor. Alternatively, you can use SetCurrentHotspotInfo ⁵⁵⁵.

AName – name of *hotspot* ¹⁷², any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW ⁵⁶⁴ method.

AlmageList – not used, reserved, set it to *nil*.

AlmageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyExEd ⁵⁶² 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 ⁵⁶² method.

ATag – *tag* of the item. You can use value returned by GetHotspotInfo ²⁹⁵ or GetItemTag ³⁰² for this item. The *tag* can also be set by SetItemTagEd ⁵⁶³ 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 ³⁵⁷).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd ⁵⁶² and SetItemExtraStrPropertyEd ⁵⁶².

When possible, use SetCurrentHotspotInfo ⁵⁵⁵ instead of this method.

See also methods of TCustomRichView:

- GetHotspotInfo ²⁹⁵;
- GetItemStyle ³⁰²;
- SetHotspotInfo ³⁵⁷.

See also methods:

- SetCurrentHotspotInfo ⁵⁵⁵.

See also properties:

- ItemCount ²³⁷.

See also:

- Modifying RichView items ¹¹²;
- Item types ⁸⁵;
- "Tags" ⁹¹.

6.1.2.2.119 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: TRVExtraItemProperty1000; 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 ItemCount²³⁷-1. Items of subdocuments (table cells⁸⁹⁵) 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 (SelSelectionBounds³⁶⁵), and call SetItemExtraIntProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraIntProperty[Ex]⁵⁵⁶.

In **SetItemExtraIntPropertyEd**, **Prop**'s type is TRVExtraItemProperty¹⁰⁰⁰. 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*** constants¹⁰⁵¹

If **AutoReformat=True**, these methods automatically reformat the document and call OnChange⁵⁷².

If **AutoReformat=False** (this can be useful if you set several properties of an item at once), call Change⁴⁹⁹ 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 BeginItemModify⁴⁹⁵ and EndItemModify⁵⁰³);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call Reformat³³²).

Methods type: **I** editing-style (unlike SetItemExtraIntProperty[Ex]³⁵⁸).

When possible, use SetCurrentItemExtraIntProperty[Ex]⁵⁵⁶ instead of these methods.

See also methods:

- SetItemExtraStrPropertyEd⁵⁶²;
- SetCurrentItemExtraIntProperty[Ex]⁵⁵⁶.

See also methods of TCustomRichView:

- SetItemExtraIntProperty[Ex]³⁵⁸.

6.1.2.2.120 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(1005); const Value: TRVUnicodeString(1032);
AutoReformat: Boolean);
procedure SetItemExtraStrPropertyExEd(ItemNo, Prop: Integer;
const Value: TRVUnicodeString(1032); 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⁽¹⁰⁰⁵⁾. 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⁽¹⁰⁵⁶⁾

ItemNo – index of the item, from 0 to ItemCount⁽²³⁷⁾-1.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽³⁶⁵⁾), and call SetItemExtraStrProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraStrProperty⁽⁵⁵⁷⁾.

If **AutoReformat=True**, these methods automatically reformat the document and call OnChange⁽⁵⁷²⁾.

If **AutoReformat=False** (this can be useful if you set several properties of an item at once), call Change⁽⁴⁹⁹⁾ 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⁽⁴⁹⁵⁾ and EndItemModify⁽⁵⁰³⁾);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call Reformat⁽³³²⁾).

Method type: **I** editing-style (unlike SetItemExtraStrProperty[Ex]⁽³⁵⁹⁾).

When possible, use SetCurrentItemExtraStrProperty[Ex]⁽⁵⁵⁷⁾ instead of this method.

See also methods:

- SetCurrentItemExtraStrProperty[Ex]⁽⁵⁵⁷⁾;
- SetItemExtraIntProperty[Ex]Ed⁽⁵⁶²⁾.

See also methods of TCustomRichView:

- SetItemExtraStrProperty[Ex]⁽³⁵⁹⁾.

6.1.2.2.121 TRichViewEdit.SetItemTagEd

Changes tag of the **ItemNo**-th item.

```
procedure SetItemTagEd(ItemNo: Integer; const ATag: TRVTag(1029));
```

Items are indexed from 0 to ItemCount⁽²³⁷⁾-1.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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⁽³⁶⁵⁾), and call SetItemTagEd of this inplace-editor. Alternatively, you can use SetCurrentTag⁽⁵⁶⁰⁾.

Tag can be an integer value or a pointer to dynamically allocated string, see *Tags*⁽⁹¹⁾.

Method type:  editing-style, unlike *SetItemTag*⁽³⁶⁰⁾.

When possible, use *SetCurrentTag*⁽⁵⁶⁰⁾ instead of this method.

See also methods of TCustomRichView:

- *GetItemTag*⁽³⁰²⁾;
- *SetItemTag*⁽³⁶⁰⁾.

See also methods:

- *SetCurrentTag*⁽⁵⁶⁰⁾.

See also properties:

- *ItemCount*⁽²³⁷⁾.

See also:

- *Modifying RichView items*⁽¹¹²⁾;
- *"Tags"*⁽⁹¹⁾.

6.1.2.2.122 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(993));
```

```
procedure SetItemTextEdW(ItemNo: Integer; const s: TRVUnicodeString(1032));
```

(Introduced in version 1.4, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to *ItemCount*⁽²³⁷⁾-1, *GetItemStyle*⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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 (*SelfSelectionBounds*⁽³⁶⁵⁾), and call *SetItemTextEd** of this inplace-editor. Alternatively, you can use *SetCurrentItemText**⁽⁵⁵⁷⁾.

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*⁽⁷⁰⁰⁾. If it is equal to *DEFAULT_CHARSET*, and for non-text items, *Style*⁽²⁵³⁾.*DefCodePage*⁽⁶³⁵⁾ is used.

SetItemTextEd works:

- like **SetItemTextEdA**, in Delphi 2007 and older
- like **SetItemTextEdW**, in Delphi 2009 and newer
- with UTF-8 string, in Lazarus⁽¹⁵⁴⁾

Methods type:  editing-style.

See also methods:

- `SetCurrentItemText -A -W` ⁽⁵⁵⁷⁾.

See also methods of TCustomRichView:

- `GetItemText -A -W` ⁽³⁰³⁾;
- `SetItemText -A -W` ⁽³⁶⁰⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also:

- [Modifying RichView items](#) ⁽¹¹²⁾;
- ["Tags"](#) ⁽⁹¹⁾;
- [Unicode in RichView](#) ⁽¹³⁰⁾.

6.1.2.2.123 TRichViewEdit.SetItemVAlignEd

Changes vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
procedure SetItemVAlignEd(ItemNo: Integer; VAlign: TRVVAlign (1033));
```

(introduced in version 12)

Items are indexed from 0 to `ItemCount` ⁽²³⁷⁾ -1.

Items of subdocuments (table cells ⁽⁸⁹⁵⁾) 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` ⁽³⁶⁵⁾), and call `SetItemVAlignEd` of this inplace-editor. Alternatively, you can use `SetCurrentItemVAlign` ⁽⁵⁵⁸⁾.

Method type: **I** editing-style, unlike `SetItemVAlign` ⁽³⁶¹⁾.

This method must not be called for text items ⁽¹⁶¹⁾, tables ⁽¹⁷³⁾, *breaks* ⁽¹⁶⁷⁾, tabs ⁽¹⁶²⁾, list markers ⁽¹⁷⁴⁾, footnotes ⁽¹⁸⁰⁾ and endnotes ⁽¹⁷⁸⁾.

When possible, use `SetCurrentItemVAlign` ⁽⁵⁵⁸⁾ instead of this method.

See also methods of TCustomRichView:

- `GetItemVAlign` ⁽³⁰³⁾;
- `SetItemVAlign` ⁽³⁶¹⁾.

See also methods:

- `SetCurrentItemVAlign` ⁽⁵⁵⁸⁾.

See also properties:

- `ItemCount` ⁽²³⁷⁾.

See also:

- [Modifying RichView items](#) ⁽¹¹²⁾.

6.1.2.2.124 TRichViewEdit.SetPictureInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a picture ⁽¹⁶³⁾ or *hot-picture* ⁽¹⁶⁶⁾.

```
procedure SetPictureInfoEd(ItemNo: Integer; const AName: TRVUnicodeString (1032));
```

```
Agr: TRVGraphic(970); AAlign: TRVAlign(1033); const ATag: TRVTag(1029));
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture⁽¹⁶³⁾ or *hot-picture*⁽¹⁶⁶⁾ type (*rsvPicture* or *rsvHotPicture*⁽¹⁰⁵⁹⁾), otherwise the method raises *ERichViewError*⁽⁹⁵⁷⁾ exception. Items are indexed from 0 to *ItemCount*⁽²³⁷⁾ -1, *GetItemStyle*⁽³⁰²⁾ returns type of item.

Items of subdocuments (table cells⁽⁸⁹⁵⁾) 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 (*SelfSelectionBounds*⁽³⁶⁵⁾), and call *SetPictureInfoEd* of this inplace-editor. Alternatively, you can use *SetCurrentPictureInfo*⁽⁵⁵⁹⁾.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using *SetItemTextEd*⁽⁵⁶⁴⁾ method.

Agr – graphic object. If this item does not use a shared image (by default), **Agr** must be a unique graphic object (you cannot assign the same graphic object to two or more items) or the value returned by *GetPictureInfo*⁽³¹⁰⁾ for this item (the image is shared if *rsvShared* extra item property⁽¹⁰⁰⁰⁾ is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by *GetPictureInfo*⁽³¹⁰⁾ for this item, the previously used graphic object is freed.

AAlign – vertical alignment of the picture.

ATag – *tag* of the item. You can use value returned by *GetPictureInfo*⁽³¹⁰⁾ or *GetItemTag*⁽³⁰²⁾ for this item. The *tag* can also be set by *SetItemTagEd*⁽⁵⁶³⁾ method.

Method type:  editing-style (unlike *SetPictureInfo*⁽³⁶³⁾).

Additional item properties are assigned by the methods *SetItemExtraIntPropertyEd*⁽⁵⁶²⁾ and *SetItemExtraStrPropertyEd*⁽⁵⁶²⁾.

When possible, use *SetCurrentPictureInfo*⁽⁵⁵⁹⁾ instead of this method.

See also methods of TCustomRichView:

- *GetPictureInfo*⁽³¹⁰⁾;
- *GetItemStyle*⁽³⁰²⁾;
- *SetPictureInfo*⁽³⁶³⁾.

See also methods:

- *SetCurrentPictureInfo*⁽⁵⁵⁹⁾.

See also properties:

- *ItemCount*⁽²³⁷⁾.

See also:

- *Modifying RichView items*⁽¹¹²⁾;
- *Item types*⁽⁸⁵⁾;
- *"Tags"*⁽⁹¹⁾.

6.1.2.2.125 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⁽¹¹⁵⁾.

See also methods:

- BeginUndoGroup⁽⁴⁹⁷⁾;
- BeginUndoCustomGroup⁽⁴⁹⁶⁾.

6.1.2.2.126 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⁽⁵⁶⁷⁾ and UndoName⁽⁵⁶⁸⁾.

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: **I** editing-style.

See also:

- Redo⁽⁵⁴⁰⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.2.127 TRichViewEdit.UndoAction

Returns type of operation to be undone⁽⁵⁶⁷⁾ next.

function UndoAction: TRVUndoType⁽¹⁰³¹⁾;

(Introduced in version 1.3)

See TRVUndoType⁽¹⁰³¹⁾ for the list of operations. If the returned value is *rvutCustom*, you can use UndoName⁽⁵⁶⁸⁾.

See also:

- RedoAction⁽⁵⁴⁰⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.2.128 TRichViewEdit.UndoName

Returns name of operation to be undone⁽⁵⁶⁷⁾ next.

function UndoName: TRVUnicodeString⁽¹⁰³²⁾;

(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⁽⁵⁶⁷⁾).

Returns non-empty string only for custom undo operation initiated by BeginUndoCustomGroup⁽⁴⁹⁶⁾.

See also:

- RedoName⁽⁵⁴⁰⁾;
- Undo in RichViewEdit⁽¹¹⁵⁾.

6.1.2.3 Events

Derived from TCustomRichViewEdit

- OnAfterOleDrop⁽⁵⁷⁰⁾ [VCL,LCL]
- OnBeforeOleDrop⁽⁵⁷⁰⁾ [VCL,LCL]
- OnCaretGetOut⁽⁵⁷¹⁾
- OnCaretMove⁽⁵⁷²⁾
- OnChange⁽⁵⁷²⁾
- OnChanging⁽⁵⁷²⁾
- OnCheckStickingItems⁽⁵⁷³⁾
- OnCurParaStyleChanged⁽⁵⁷³⁾
- OnCurTextStyleChanged⁽⁵⁷⁴⁾
- OnDrawCurrentLine⁽⁵⁷⁴⁾
- OnDrawCustomCaret⁽⁵⁷⁵⁾ [VCL,LCL]
- OnDropFile⁽⁵⁷⁶⁾
- OnDropFiles⁽⁵⁷⁶⁾
- OnItemTextEdit⁽⁵⁷⁸⁾
- OnMeasureCustomCaret⁽⁵⁷⁸⁾
- OnOleDragEnter⁽⁵⁷⁹⁾ [VCL,LCL]
- OnOleDragLeave⁽⁵⁷⁹⁾ [VCL,LCL]
- OnOleDragOver⁽⁵⁸⁰⁾ [VCL,LCL]
- OnOleDrop⁽⁵⁸¹⁾ [VCL,LCL]
- OnParaStyleConversion⁽⁵⁸²⁾
- OnPaste⁽⁵⁸⁴⁾
- OnSaveCustomFormat⁽⁵⁸⁵⁾
- OnSmartPopupClick⁽⁵⁸⁵⁾
- OnStyleConversion⁽⁵⁸⁵⁾

Derived from TCustomRichView⁽²¹⁰⁾

- OnAddStyle⁽³⁷⁰⁾
- OnAfterDrawImage⁽³⁷⁰⁾
- OnAssignImageFileName⁽³⁷¹⁾
- OnCheckpointVisible⁽³⁷²⁾ (doesn't work in editor)

- OnCMHintShow³⁷² [VCL,LCL]
- OnControlAction³⁷³
- OnCopy³⁷⁴
- OnGetItemCursor³⁷⁴
- OnGetSpellingSuggestions³⁷⁵ [FMX]
- OnHTMLSaveImage³⁷⁵
- OnImportFile³⁷⁸
- OnImportPicture³⁷⁸
- OnItemAction³⁸⁰
- OnItemHint³⁸¹
- OnJump³⁸²
- OnLoadCustomFormat³⁸³
- OnLoadDocument³⁸⁴
- OnNewDocument³⁸⁴
- OnPaint³⁸⁴
- OnProgress³⁸⁵
- OnReadField³⁸⁶
- OnReadHyperlink³⁸⁸
- OnReadMergeField³⁹⁰
- OnRVDbClick³⁹¹
- OnRVFControlNeeded³⁹²
- OnRVFImageListNeeded³⁹³
- OnRVFPictureNeeded³⁹³
- OnRVMouseDown³⁹⁴
- OnRVMouseMove³⁹⁵
- OnRVMouseUp³⁹⁶
- OnRVRightClick³⁹⁷
- OnSaveComponentToFile³⁹⁷
- OnSaveDocXExtra³⁹⁹
- OnSaveHTMLExtra⁴⁰¹
- OnSaveImage2⁴⁰²
- OnSaveItemToFile⁴⁰⁴
- OnSaveParaToHTML⁴⁰⁶
- OnSaveRTFExtra⁴⁰⁶
- OnSelect⁴⁰⁸
- OnSpellingCheck⁴⁰⁹
- OnStyleTemplatesChange⁴¹⁰
- OnTextFound⁴¹⁰
- OnWriteHyperlink⁴¹¹
- OnWriteObjectProperties⁴¹³

Derived from TRVScroller⁸¹³

- OnHScrolled⁸²¹
- OnVScrolled⁸²¹

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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 OnAfterOleDrop: TNotifyEvent;
```

(introduced in version 16)

When the editor accepts dropped data, the following sequence of actions happens:

1. **OnBeforeOleDrop** event
2. data insertion (OnOleDrop⁵⁸¹ 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'⁽¹¹⁸⁾;

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...
<i>rvdLeft</i>	The user pressed Left (or Right , if BiDiMode ⁽²²⁴⁾ = <i>rvbdRightToLeft</i>) while the caret is at the beginning of the document
<i>rvdUp</i>	The user pressed Up while the caret is in the first line of the document
<i>rvdRight</i>	The user pressed Right (or Left , if BiDiMode ⁽²²⁴⁾ = <i>rvbdRightToLeft</i>) while the caret is at the end of the document
<i>rvdDown</i>	The user pressed Down while the caret is in the last line of the document.
<i>rvdTop</i>	The user pressed Ctrl *+ Home

Direction	The event happens if...
<code>rvdBottom</code>	The user pressed <code>Ctrl</code> *+ <code>End</code>

* 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`⁽³⁴⁹⁾ here). If you need to update user interface reflecting selection, use `OnSelect`⁽⁴⁰⁸⁾.

In this event, you can use the following:

- `CurItemNo`⁽⁴⁷²⁾ property;
- `OffsetInCurItem`⁽⁴⁷⁹⁾ property;
- `GetCurrentLineCol`⁽⁵¹²⁾ 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`⁽⁴⁹⁹⁾.

 Viewer-style methods do not generate this event.

See also property:

- `Modified`⁽⁴⁷⁹⁾.

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 **I** editing-style method (before changes are actually done).

 Viewer-style methods do not generate this event.

See also:

- **OnChange**⁵⁷² event;
- **ReadOnly**⁴⁸⁰ property.

6.1.2.3.6 TRichViewEdit.OnCheckStickingItems

Allows to "stuck" items together.

type

```
TRVStickingItemsEvent = procedure (Sender: TCustomRichViewEdit;  
  RVData: TCustomRVData957; 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**⁷⁰⁵ property of a text style).

6.1.2.3.7 TRichViewEdit.OnCurParaStyleChanged

Occurs when the index of current paragraph style (value of **CurParaStyleNo**⁴⁷³ 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**⁴⁷³.

See also events:

- **OnCurTextStyleChanged**⁵⁷⁴.

6.1.2.3.8 TRichViewEdit.OnCurTextStyleChanged

Occurs when the index of current text style (value of `CurTextStyleNo`⁽⁴⁷⁴⁾ 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`⁽⁴⁷⁴⁾.

See also events:

- `OnCurParaStyleChanged`⁽⁵⁷³⁾.

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(461);
  ACanvas: TCanvas; ARVData: TCustomRVFormattedData(957);
  AStartItemNo, AStartOffs, AEndItemNo, AEndOffs: Integer;
  const ARect: TRVCoordRect(998); APrepaint: Boolean) of object;
```

property `OnDrawCurrentLine`: `TRVDrawCurrentLineEvent`;

FMX:

```
TRVDrawCurrentLineEvent = procedure(Sender: TCustomRichViewEdit(461);
  ACanvas: TRVFMXCanvas(961); ARVData: TCustomRVFormattedData(957);
  AStartItemNo, AStartOffs, AEndItemNo, AEndOffs: Integer;
  const ARect: TRVCoordRect(998); APrepaint: Boolean) of object;
```

property `OnDrawCurrentLine`: `TRVDrawCurrentLineEvent`;

(introduced in version 21)

Parameters:

Canvas – canvas to paint.

ARVData – document containing the current line (`Sender.RVData`⁽²⁴⁷⁾)

AStartItemNo – index of the first item on this line (in the range from 0 to `ARVData.ItemCount`⁽²³⁷⁾ - 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`⁽²³⁷⁾ - 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: TCustomRichViewEdit461;
    Canvas: TCanvas; const Rect: TRVCoordRect998) of object;
property OnDrawCustomCaret: TRVDrawCustomCaretEvent;
(introduced in version 10)

```

Custom caret is drawn if CustomCaretInterval⁴⁷⁴>0.

OnMeasureCustomCaret⁵⁷⁸ event occurs before each call of this event.

Example

This example draws a rectangle/ellipse caret. In combination with the example in OnMeasureCustomCaret⁵⁷⁸ topic, it draws rectangle/ellipse around the character to the right of the caret.

```

procedure TMyForm.MyRichViewEditDrawCustomCaret(
  Sender: TCustomRichViewEdit;
  Canvas: TCanvas; const Rect: TRVCoordRect998);
const f: Boolean=True;
var r: TRVCoordRect998;
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: TRVUnicodeString1032; 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 PasteGraphicFiles⁵³⁵), or as a result of drag&drop¹¹⁸.

In this event, you can insert content of this file in the caret position.

Input parameters:

FileName – 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 AcceptDragDropFormats⁴⁷⁰ 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` ⁽⁵⁷⁶⁾ 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) ⁽⁵⁹³⁾.

See also:

- `Drag&Drop` ⁽¹¹⁸⁾.

6.1.2.3.13 TRichViewEdit.OnItemResize

Occurs after resizing image ⁽¹⁶³⁾, control ⁽¹⁶⁸⁾ or table ⁽¹⁷³⁾ with the mouse.

type

```
TRVItemResizeEvent = procedure(Sender: TCustomRichViewEdit;
  RVData: TCustomRVFormattedData (957); 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` ⁽²⁴⁷⁾, table cell ⁽⁸⁹⁵⁾, or `RVData` of cell inplace-editor. **ItemNo** – index of this item inside **RVData**.

(**Val1**, **Val2**) are:

- for tables ⁽¹⁷³⁾: (0, row index) after resizing row, or (1, column index) after resizing columns;
- for controls ⁽¹⁶⁸⁾: (old control width, old control height);
- for other resizable items (pictures ⁽¹⁶³⁾, shapes in Report Workshop): (old value of `rveplImageWidth` ⁽¹⁰⁰⁰⁾ property, old value of `rveplImageHeight` ⁽¹⁰⁰⁰⁾ property).

6.1.2.3.14 TRichViewEdit.OnItemTextEdit

Occurs when text item⁽¹⁶¹⁾'s text is changed as a result of editing operation.

type

```
TRVItemTextEditEvent = procedure (Sender: TCustomRichViewEdit;
  const OldText: TRVUnicodeString(1032); RVData: TCustomRVData(957);
  ItemNo: Integer; var NewTag: TRVTag(1029); 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**⁽²⁴⁷⁾, table cell⁽⁸⁹⁵⁾, or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**. You can get text after the editing operation using **RVData.GetItemTextW**⁽³⁰³⁾ (**ItemNo**).

NewTag – the item tag⁽⁹¹⁾, the same as **RVData.GetItemTag**⁽³⁰²⁾ (**ItemNo**)

NewStyleNo – the item style, index in the collection **Style**⁽²⁵³⁾.**TextStyles**⁽⁶⁵⁴⁾, the same as **RVData.GetItemStyle**⁽³⁰²⁾ (**ItemNo**)

Output Parameters:

NewTag – value of tag⁽⁹¹⁾ 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(998)) of object;
property OnMeasureCustomCaret: TRVMeasureCustomCaretEvent;
```

(introduced in version 10)

Custom caret [VCL and LCL]

A custom caret is drawn if CustomCaretInterval⁽⁴⁷⁴⁾>0.

This event is called before each call to OnDrawCustomCaret⁽⁵⁷⁵⁾. 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⁽⁵⁷⁵⁾.

Standard caret

If the global variable `RichViewEditCustomCaretSize1040 = True`, this even is called for the standard caret as well. If `RichViewEditCaretPosition1040 <> 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: TCustomRichView210;  
  const DataObject: IDataObject; Shift: TShiftState;  
  X, Y: Integer; PossibleDropEffects: TRVOleDropEffects1015;  
  var DropEffect: TRVOleDropEffect1015) 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, `OnOleDragOver580` will be called while the user moves the mouse pointer inside the editor. Dragging will be finished either by calling `OnOleDrop581` (successfully) or by calling `OnOleDragLeave579` (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:

- `OnOleDragOver580`;
- `OnOleDragLeave579`;
- `OnOleDrop581`.

See also:

- Drag&drop in `TRichView118`.

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`⁵⁷⁹, it occurs when the user moves the mouse pointer into the editor while dragging. After that, `OnOleDragOver`⁵⁸⁰ is called while the user moves the mouse inside the editor. Dragging is finished either by calling `OnOleDrop`⁵⁸¹ (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`⁵⁷⁹;
- `OnOleDragOver`⁵⁸⁰;
- `OnOleDrop`⁵⁸¹.

See also:

- Drag&drop in TRichView¹¹⁸.

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`⁵⁷⁹ (or automatically by the editor).

type

```
TRVOleDragOverEvent = procedure (Sender: TCustomRichView210;
  Shift: TShiftState; X, Y: Integer;
  PossibleDropEffects: TRVOleDropEffects1015;
  var DropEffect: TRVOleDropEffect1015) 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`⁵⁷⁹ occurs before this event, with DataObject in parameters. Dragging will be finished either by calling `OnOleDrop`⁵⁸¹ (successfully) or by calling `OnOleDragLeave`⁵⁷⁹ (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⁽⁵⁷⁹⁾;
- OnOleDragLeave⁽⁵⁷⁹⁾;
- OnOleDrop⁽⁵⁸¹⁾.

See also:

- Drag&drop in TRichView⁽¹¹⁸⁾.

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⁽⁵⁷⁹⁾, it occurs when the user moves the mouse pointer into the editor while dragging. After that, OnOleDragOver⁽⁵⁸⁰⁾ is called while the user moves the mouse inside the editor. Dragging is finished either by calling this event (successfully) or by calling OnOleDragLeave⁽⁵⁷⁹⁾ (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`⁽⁵⁷⁰⁾ events.

See also events:

- `OnOleDragEnter`⁽⁵⁷⁹⁾;
- `OnOleDragOver`⁽⁵⁸⁰⁾;
- `OnOleDragLeave`⁽⁵⁷⁹⁾;
- `OnDropFiles`⁽⁵⁷⁶⁾.

See also:

- Drag&drop in TRichView'⁽¹¹⁸⁾;
- How to make one line, plain text editor⁽¹⁰⁶⁶⁾.

6.1.2.3.20 TRichViewEdit.OnParaStyleConversion

Occurs while executing `ApplyParaStyleConversion`⁽⁴⁹¹⁾ 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`⁽²⁵³⁾.`ParaStyles`⁽⁶⁴⁸⁾);

UserData – value passed as a parameter in `ApplyParaStyleConversion`⁽⁴⁹¹⁾;

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(718);  
begin  
  // creating paragraph style with the desired attributes  
  ParaInfo := TParaInfo.Create(nil);  
  try  
    ParaInfo.Assign(RVStyle1.ParaStyles(648)[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(661)(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⁽⁴⁹¹⁾ as UserData;
- RVStyle1 – TRVStyle⁽⁶³⁰⁾ 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, Standard⁶⁹⁵ property is set to *False*. It is important, otherwise DeleteUnusedStyles²⁸⁶ 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:

- OnStyleConversion⁵⁸⁵.

See also methods:

- ApplyParaStyleConversion⁴⁹¹.

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 (Paste⁵³⁴ 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 Paste⁵³⁴ 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.InsertText532(s,False);
  DoDefault := False;
end;
```

See also:

- RichView and Clipboard¹⁰⁸;
- How to make a plain text editor¹⁰⁶⁶.

6.1.2.3.22 TRichViewEdit.OnSaveCustomFormat

This event allows saving data to field in your own format.

type

```
TRVCustomFormatEvent = procedure (Sender: TCustomRichView210;  
    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 Document²³² property)
- by TDBRichViewEdit⁶⁰⁶ 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) Document²³².FieldFormat⁴²⁴
- (for TDBRichViewEdit⁶⁰⁶) FieldFormat⁶¹³ property.

See also events:

- OnLoadCustomFormat³⁸³.

6.1.2.3.23 TRichViewEdit.OnSmartPopupClick

Occurs when user clicks "smart popup"⁵⁸⁸ 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 ShortCut⁵⁹¹). If you use menu, it's not necessary to process this event.

6.1.2.3.24 TRichViewEdit.OnStyleConversion

Occurs while executing ApplyStyleConversion⁴⁹² 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(253).TextStyles(654)`);

UserData – value passed as a parameter in `ApplyStyleConversion(492)`;

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(474)`);

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(705))) then
    NewStyleNo := UserData;
end;

```

This example does exactly the same work as `ApplyTextStyle(494)`, i.e.

`MyRichViewEdit.ApplyStyleConversion(492)(StyleNo)` will be equivalent to `MyRichViewEdit.ApplyTextStyle(494)(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(712);
begin
  // creating text style with the desired attributes
  FontInfo := TFontInfo.Create(nil);
  try
    FontInfo.Assign(RVStyle1.TextStyles(654)[StyleNo]);
    case UserData of
      TEXT_BOLD:
        if btnBold.Down then
          FontInfo.Style := FontInfo.Style(708)+[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.FindTextStyle661(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 ApplyStyleConversion⁴⁹² as UserData;
- RVStyle1 – TRVStyle⁶³⁰ 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:

- OnParaStyleConversion⁵⁸².

See also methods:

- ApplyStyleConversion⁴⁹².

6.1.2.4 Classes of properties

TRVSmartPopupProperties⁽⁵⁸⁸⁾ is a type of TRichViewEdit⁽⁴⁶¹⁾.SmartPopupProperties⁽⁴⁸⁰⁾ 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⁽⁴⁸⁰⁾.

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⁽⁸⁵⁾. 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⁽⁵⁷²⁾ event to set properties of "smart popup" and display the button. The button is displayed when you assign True to TCustomRichViewEdit.SmartPopupVisible⁽⁴⁸¹⁾. 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⁽⁵⁸⁹⁾;
- Color⁽⁵⁸⁹⁾, LineColor⁽⁵⁹¹⁾;
- HoverColor⁽⁵⁹⁰⁾, HoverLineColor⁽⁵⁹⁰⁾;
- ImageList⁽⁵⁹⁰⁾, ImageIndex⁽⁵⁹⁰⁾.

Hint⁽⁵⁹⁰⁾ is displayed when the user moves the mouse pointer over the button.

Position⁽⁵⁹¹⁾ defines the button position.

When the user clicks the button or presses ShortCut⁽⁵⁹¹⁾, Menu⁽⁵⁹¹⁾ or Popup⁽⁵⁹¹⁾ is displayed (if assigned). Instead of a menu/popup, you can process TCustomRichViewEdit.OnSmartPopupClick⁽⁵⁸⁵⁾ event to execute a command or display a dialog.

6.1.2.4.1.1 Properties

In TRVSmartPopupProperties

- ButtonType⁽⁵⁸⁹⁾
- Color⁽⁵⁸⁹⁾

- Hint ⁵⁹⁰
- HoverColor ⁵⁹⁰
- HoverLineColor ⁵⁹⁰
- ImageIndex ⁵⁹⁰
- ImageList ⁵⁹⁰
- LineColor ⁵⁹¹
- Menu ⁵⁹¹
- Popup ⁵⁹¹ [FMX]
- Position ⁵⁹¹
- ShortCut ⁵⁹¹

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
<i>rvsptDropDown</i>	Button width is increased, a triangle is shown. Recommended for buttons displaying menu ⁵⁹¹ or popup windows.	
<i>rvsptShowDialog</i>	Button width is increased, an ellipsis is shown. Recommended for buttons displaying dialogs.	
<i>rvsptSimple</i>	Button width is not changed. Recommended for buttons executing an immediate command.	

You can change this property in TCustomRichViewEdit.OnCaretMove ⁵⁷² (to create the proper button for the item at the position of caret)

Default value:

rvsptDropDown

Background color of a button.

property Color: TRVColor ⁹⁹⁶;

Default value:

rvclWindow ¹⁰³⁸

See also:

- LineColor ⁵⁹¹;
- HoverColor ⁵⁹⁰;
- HoverLineColor ⁵⁹⁰.

Contains the text string that can appear when the user moves the mouse over a button.

```
property Hint: TRVUnicodeString1032;
```

You can change this property in `TCustomRichViewEdit.OnCaretMove`⁵⁷² (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: TRVColor996;
```

Default value:

`rvclInfoBk`¹⁰³⁸

See also:

- `Color`⁵⁸⁹;
- `LineColor`⁵⁹¹;
- `HoverLineColor`⁵⁹⁰.

Color of button border and the right side sign (ellipsis or triangle)⁵⁸⁹ when a button is under the mouse pointer.

```
property HoverLineColor: TRVColor996;
```

Default value:

`rvclInfoText`¹⁰³⁸

See also:

- `Color`⁵⁸⁹;
- `HoverColor`⁵⁹⁰;
- `HoverColor`⁵⁹⁰.

Index of a button image in `ImageList`⁵⁹⁰.

```
property ImageIndex: Integer;
```

This image is displayed on a button.

You can change this property in `TCustomRichViewEdit.OnCaretMove`⁵⁷² (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`⁵⁹⁰-th image is shown on button.

You can change this property in `TCustomRichViewEdit.OnCaretMove`⁵⁷² (to create the proper button for the item at the position of caret).

Border color of a button.

```
property LineColor: TRVColor996;
```

Default value:

*rvclHighlight*¹⁰³⁸

See also:

- Color⁵⁸⁹;
- HoverColor⁵⁹⁰;
- HoverLineColor⁵⁹⁰.

A link to a popup menu to display when the user clicks a button or presses ShortCut⁵⁹¹.

```
property Menu: TPopupMenu;
```

Alternatively, you can:

- process TCustomRichViewEdit.OnSmartPopupClick⁵⁸⁵ event to execute a command or display a dialog;
- (in FireMonkey) assign Popup⁵⁹¹ property.

You can change this property in TCustomRichViewEdit.OnCaretMove⁵⁷² (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 Popup⁵⁹¹ and **Menu** are assigned, Popup⁵⁹¹ is displayed.

A link to TPopup component to display when the user clicks a button or presses ShortCut⁵⁹¹.

```
property Popup: TPopup;
```

Alternatively, you can:

- process TCustomRichViewEdit.OnSmartPopupClick⁵⁸⁵ event to execute a command or display a dialog;
- assign Menu⁵⁹¹ property.

You can change this property in TCustomRichViewEdit.OnCaretMove⁵⁷² (to create the proper button for the item at the position of the caret).

If both **Popup** and Menu⁵⁹¹ 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⁽⁵⁹¹⁾.

See also events of TCustomRichViewEdit:

- OnSmartPopupClick⁽⁵⁸⁵⁾.

6.1.2.4.1.2 Methods

In TRVSmartPopupProperties

SetButtonState⁽⁵⁹²⁾

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⁽⁵⁸⁵⁾.

6.1.2.5 Examples

TRichViewEdit examples

- Moving caret to the beginning of paragraph⁽⁵⁹²⁾
- Using OnDropFiles to accept files of different formats⁽⁵⁹³⁾

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(461);
  ParagraphIndex: Integer);
var i: Integer;
begin
  for i := 0 to rve.ItemCount(237)-1 do
    begin
      if rve.IsParaStart(318)(i) then
        dec(ParagraphIndex);
      if ParagraphIndex<0 then begin
        rve.SetSelectionBounds(365)(i, rve.GetOffsBeforeItem(309)(i),
          i, rve.GetOffsBeforeItem(309)(i));
        rve.Invalidate;
        exit;
      end;
    end;
  end;
end;
```

Call:

```
GoToParagraph(RichViewEdit1, 7);
```

ParagraphIndex is zero-based.

See also:

- TCustomRichViewEdit⁴⁶¹.MoveCaret⁵³⁴

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 OnDropFiles⁵⁷⁶ 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:TRVOfficeConverter⁷⁹⁵ 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: TRVGraphic970;
  Ext: String;
  i: Integer;
begin
  Result := False;
  try
    // 1. Trying to insert as a graphic
    gr := RVGraphicHandler1057.LoadFromFile(FileName);
    if gr <> nil then
      begin
        Sender.InsertPicture526(' ', gr, rvvaBaseline1033);
        Result := True;
        exit;
      end;
    // 2. Trying to insert as RTF file
    Ext := LowerCase(ExtractFileExt(FileName));
    if Ext='.rtf' then
      begin
        Sender.InsertRTFFromFileEd527(FileName);
        Result := True;
        exit;
      end;
    // 3. Trying to insert as RVF file
    if Ext='.rvf' then
```

```

begin
  Sender.InsertRVFFFromFileEd528(FileName);
  Result := True;
  exit;
end;
// 4. Trying to insert as text file
if Ext='.txt' then
begin
  Sender.InsertTextFromFile533(FileName);
  Result := True;
  exit;
end;
// 5. Trying to insert using office converters
for i := 0 to RVOOfficeConverter1.ImportConverters798.Count-1 do
  if Pos(Ext, RVOOfficeConverter1.ImportConverters[i].Filter)>0 then
    if RVOOfficeConverter1.ImportRTF801(FileName, i) then
      begin
        RVOOfficeConverter1.Stream799.Position := 0;
        Sender.InsertRTFFromFileEd527(RVOOfficeConverter1.Stream);
        RVOOfficeConverter1.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;
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 Style²⁵³ property.

The most important property settings can be done in the component editor²¹⁸ 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 TRichView²¹⁰ instead, to conserve system resources.

For editing documents in database fields, use TDBRichViewEdit⁶⁰⁶.

In Delphi XE2 or newer, there is another solution for data awareness: LiveBindings (see `TRichView.Document`⁽²³²⁾ property).

Hierarchy

TObject

TPersistent

TComponent

TControl

TWinControl

TCustomControl

TRVScroller⁽⁸¹³⁾

TCustomRichView⁽²¹⁰⁾

Field Type

This component must be linked to BLOB field.

If you use RVF⁽¹²⁴⁾ (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`⁽⁶⁰⁰⁾.`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`⁽²⁴⁷⁾, the component clears `BackgroundBitmap`⁽²²²⁾ before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF⁽¹²⁴⁾ (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`⁽³⁸³⁾ 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`⁽⁵⁹⁸⁾, 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⁽³⁸⁴⁾ 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⁽³⁸⁴⁾ occurs.

OnLoadCustomFormat⁽³⁸³⁾ allows loading documents in custom formats.

6.1.3.1 Properties

In TDBRichView

- AllowMarkdown⁽⁵⁹⁸⁾
- AutoDeleteUnusedStyles⁽⁵⁹⁹⁾
- AutoDisplay⁽⁵⁹⁹⁾
- CodePage⁽⁵⁹⁹⁾
- DataField⁽⁶⁰⁰⁾
- DataSource⁽⁶⁰⁰⁾
- ▶ Field⁽⁶⁰⁰⁾

Derived from TCustomRichView

- AllowSelection⁽²²²⁾ (deprecated)
- AnimationMode⁽²²²⁾
- BackgroundBitmap⁽²²²⁾
- BackgroundStyle⁽²²³⁾
- BiDiMode⁽²²⁴⁾
- BottomMargin⁽²²⁵⁾
- ClearLeft⁽²²⁵⁾
- ClearRight⁽²²⁶⁾
- Color⁽²²⁶⁾
- CPEventKind⁽²²⁷⁾
- Cursor⁽²²⁷⁾
- DarkMode⁽²²⁸⁾
- Delimiters⁽²²⁸⁾
- DocObjects⁽²²⁹⁾
- DocParameters⁽²³⁰⁾
- DocProperties⁽²³¹⁾
- ▶ DocumentHeight⁽²³³⁾
- DocumentPixelsPerInch⁽²³³⁾
- DoInPaletteMode⁽²³⁴⁾
- ▶ FirstItemVisible⁽²³⁶⁾
- FirstJumpNo⁽²³⁶⁾

- HTMLReadProperties ²³⁷
- HTMLSaveProperties ²³⁷
- ▶ ItemCount ²³⁷
- ▶ LastItemVisible ²³⁸
- LeftMargin ²³⁸
- MarkdownProperties ²³⁹
- MaxLength ²³⁹
- MaxTextWidth ²³⁹
- MinTextWidth ²⁴⁰
- NoteText ²⁴⁰
- Options ²⁴⁰
- PageBreaksBeforeItems ²⁴⁴
- RightMargin ²⁴⁴
- RTFOptions ²⁴⁵
- RTFReadProperties ²⁴⁶
- ▶ RVData ²⁴⁷
- RVFOptions ²⁴⁷
- RVFParaStylesReadMode ²⁵¹
- RVFTextStylesReadMode ²⁵¹
- ▶ RVFWarnings ²⁵¹
- SingleClick ²⁵³ (deprecated)
- SelectionHandlesVisible ²⁵³ (D2010+)
- **Style** ²⁵³
- StyleTemplateInsertMode ²⁵⁴
- TabNavigation ²⁵⁵
- TopMargin ²⁵⁵
- UseStyleTemplates ²⁵⁶
- UseVCLThemes ²⁵⁷
- VAlign ²⁵⁷
- VSmallStep ²⁵⁷
- WordWrap ²⁵⁸

Derived from TRVScroller ⁸¹³

- BorderStyle ⁸¹⁶
- ▶ HScrollMax ⁸¹⁶
- HScrollPos ⁸¹⁶
- HScrollVisible ⁸¹⁶
- ▶ InplaceEditor ⁸¹⁷
- NoVScroll ⁸¹⁷
- Tracking ⁸¹⁸
- UseXPThemes ⁸¹⁸
- ▶ VScrollMax ⁸¹⁸
- VScrollPos ⁸¹⁹
- VScrollVisible ⁸¹⁹
- WheelStep ⁸¹⁹

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

- CodePage⁵⁹⁹
- TRichView²¹⁰.Document⁴²².AllowMarkdown⁴²² (for LiveBindings)

6.1.3.1.2 TDBRichView.AutoDeleteUnusedStyles

If this property is set to *True*, the component deletes unused text⁶⁵⁴, paragraph⁶⁴⁸ and list⁶⁴⁶ 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 conditions are required to auto-delete styles correctly:

- this component must not share TRVStyle⁶³⁰ component with other TRichView controls (others TRichView controls must not be linked to the same TRVStyle component via Style²⁵³ property) and

Default value:

False

See also

- TRichView²¹⁰.Document²³².AutoDeleteUnusedStyles⁴²³ (for live binding)
- TDBRichViewEdit⁶⁰⁶.AutoDeleteUnusedStyles⁶¹¹

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 LoadField⁶⁰⁴ 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: TRVCodePage996;
```

(introduced in version 23)

The default value of this property is *RV_CP_DEFAULT*¹⁰⁵⁰. 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`⁽¹⁰⁵⁰⁾

See also

- `TRichView`⁽²¹⁰⁾.`Document`⁽⁴²²⁾.`CodePage`⁽⁴²³⁾ (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`⁽⁶⁰⁰⁾ 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`⁽⁶⁰⁰⁾ 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`⁽¹²⁴⁾ (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`⁽²⁴⁷⁾).

6.1.3.2 Methods

In TDBRichView

`Create`⁽⁶⁰⁴⁾

`Destroy`⁽⁶⁰⁴⁾

LoadField⁽⁶⁰⁴⁾

Derived from TCustomRichView

-  AddBreak⁽²⁶²⁾
-  AddBullet⁽²⁶³⁾
-  AddCheckpoint⁽²⁶⁴⁾
-  AddControl⁽²⁶⁵⁾
-  AddFmt⁽²⁶⁶⁾
-  AddHotPicture⁽²⁶⁷⁾
-  AddHotspot⁽²⁶⁸⁾
-  AddItem⁽²⁶⁹⁾
-  AddNL -A -W⁽²⁷⁰⁾
-  AddPicture⁽²⁷²⁾
-  AddTab⁽²⁷³⁾
-  AddTextNL -A -W⁽²⁷⁴⁾
-  AppendFrom⁽²⁷⁵⁾
-  AppendRVFFromStream⁽²⁷⁶⁾
- AssignSoftPageBreaks⁽²⁷⁷⁾
- BeginOleDrag⁽²⁷⁸⁾
-  Clear⁽²⁷⁹⁾
- ClearLiveSpellingResults⁽²⁷⁹⁾
- ClearSoftPageBreaks⁽²⁷⁹⁾
- ClientToDocument, DocumentToClient⁽²⁸⁰⁾
- ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits⁽²⁸⁰⁾
- Copy⁽²⁸¹⁾
- CopyDef⁽²⁸¹⁾
- CopyImage⁽²⁸¹⁾
- CopyRTF⁽²⁸²⁾
- CopyRVF⁽²⁸²⁾
- CopyText -A -W⁽²⁸³⁾
- Create⁽²⁸³⁾
-  DeleteItems⁽²⁸³⁾
- DeleteMarkedStyles⁽²⁸⁴⁾
- DeleteParas⁽²⁸⁵⁾
-  DeleteSection⁽²⁸⁵⁾
- DeleteUnusedStyles⁽²⁸⁶⁾
- Deselect⁽²⁸⁶⁾
- Destroy⁽²⁸⁶⁾
- FindCheckPointByName⁽²⁸⁷⁾
- FindCheckPointByTag⁽²⁸⁷⁾
- FindControlItemNo⁽²⁸⁷⁾
- Format⁽²⁸⁸⁾
- FormatTail⁽²⁸⁸⁾
- GetBreakInfo⁽²⁸⁹⁾
- GetBulletInfo⁽²⁸⁹⁾
- GetCheckpointByNo⁽²⁹¹⁾
- GetCheckpointInfo⁽²⁹¹⁾

GetCheckpointItemNo²⁹¹
GetCheckpointNo²⁹²
GetCheckpointXY²⁹²
GetCheckpointY²⁹²
GetCheckpointYEx²⁹³
GetControlInfo²⁹³
GetFirstCheckpoint²⁹⁴
GetFocusedItem²⁹⁵
GetHotspotInfo²⁹⁵
GetItem²⁹⁶
GetItemAt²⁹⁷
GetItemCheckpoint²⁹⁸
GetItemCoords²⁹⁸
GetItemCoordsEx²⁹⁹
GetItemExtraIntProperty³⁰⁰
GetItemExtraIntPropertyEx³⁰⁰
GetItemExtraStrProperty³⁰⁰
GetItemExtraStrPropertyEx³⁰⁰
GetItemNo³⁰¹
GetItemPara³⁰¹
GetItemStyle³⁰²
GetItemTag³⁰²
GetItemText -A -W³⁰³
GetItemVAlign³⁰³
GetJumpPointLocation³⁰⁴
GetJumpPointY³⁰⁵
GetLastCheckpoint³⁰⁵
GetLineNo³⁰⁶
GetListMarkerInfo³⁰⁷
GetNextCheckpoint³⁰⁷
GetOffsAfterItem³⁰⁸
GetOffsBeforeItem³⁰⁹
GetPictureInfo³¹⁰
GetPrevCheckpoint³¹¹
GetRealDocumentPixelsPerInch³¹¹
GetSelectedImage³¹²
GetSelectionBounds³¹²
GetSelText -W³¹³
GetTextInfo³¹³
GetWordAt -A -W³¹⁴
 InsertRVFFromStream³¹⁶
IsFromNewLine³¹⁷
IsParaStart³¹⁸
LiveSpellingValidateWord³¹⁸
 LoadDocX³¹⁸ (D2009+ or with Delphi ZLib)
 LoadDocXFromStream³¹⁹ (D2009+ or with Delphi ZLib)
 LoadFromFile³²⁰

-  [LoadFromFileEx](#) ³²⁰
-  [LoadFromStream](#) ³²¹
-  [LoadFromStreamEx](#) ³²¹
-  [LoadHTML](#) ³²²
-  [LoadHTMLFromStream](#) ³²⁴
-  [LoadMarkdown](#) ³²⁵
-  [LoadMarkdownFromStream](#) ³²⁵
-  [LoadRTF](#) ³²⁶
-  [LoadRTFFromStream](#) ³²⁷
-  [LoadRVF](#) ³²⁸
-  [LoadRVFFromStream](#) ³²⁹
-  [LoadText -W](#) ³³⁰
-  [LoadTextFromStream -W](#) ³³¹
- [MarkStylesInUse](#) ³³¹
- [RefreshListMarkers](#) ³³²
- [Reformat](#) ³³²
-  [RemoveCheckpoint](#) ³³²
- [ResetAnimation](#) ³³³
- [SaveDocX](#) ³³³
- [SaveDocXToStream](#) ³³⁴
- [SaveHTML](#) ³³⁵
- [SaveHTMLToStream](#) ³³⁸
- [SavePicture](#) ³⁴²
- [SaveRTF](#) ³⁴³
- [SaveRTFToStream](#) ³⁴⁴
- [SaveRVF](#) ³⁴⁴
- [SaveRVFToStream](#) ³⁴⁴
- [SaveText -W](#) ³⁴⁵
- [SaveTextToStream -W](#) ³⁴⁵
- [SearchText -A -W](#) ³⁴⁶
- [SelectAll](#) ³⁴⁸
- [SelectControl](#) ³⁴⁸
- [SelectionExists](#) ³⁴⁹
- [SelectWordAt](#) ³⁴⁹
- [SetAddParagraphMode](#) ³⁵⁰
-  [SetBreakInfo](#) ³⁵¹
-  [SetBulletInfo](#) ³⁵²
-  [SetCheckpointInfo](#) ³⁵³
-  [SetControlInfo](#) ³⁵⁴
- [SetFooter](#) ³⁵⁵
- [SetHeader](#) ³⁵⁶
-  [SetHotspotInfo](#) ³⁵⁷
-  [SetItemExtraIntProperty](#) ³⁵⁸
-  [SetItemExtraIntPropertyEx](#) ³⁵⁸
-  [SetItemExtraStrProperty](#) ³⁵⁹
-  [SetItemExtraStrPropertyEx](#) ³⁵⁹

-  [SetItemTag](#) ⁽³⁶⁰⁾
-  [SetItemText -A -W](#) ⁽³⁶⁰⁾
-  [SetItemVAlign](#) ⁽³⁶¹⁾
-  [SetListMarkerInfo](#) ⁽³⁶²⁾
-  [SetPictureInfo](#) ⁽³⁶³⁾
- [SetSelectionBounds](#) ⁽³⁶⁵⁾
- [StartAnimation](#) ⁽³⁶⁶⁾
- [StartLiveSpelling](#) ⁽³⁶⁷⁾
- [StopAnimation](#) ⁽³⁶⁷⁾
- [StoreSearchResult](#) ⁽³⁶⁷⁾
- [UpdatePalettInfo](#) ⁽³⁶⁸⁾

Derived from TRVScroller ⁽⁸¹³⁾

[ScrollTo](#) ⁽⁸²⁰⁾

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 ⁽²¹⁸⁾ (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 ⁽⁶⁰⁴⁾ 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 ⁽⁵⁹⁹⁾ 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 ³⁷⁰
- OnAfterDrawImage ³⁷⁰
- OnAssignImageFileName ³⁷¹
- OnCheckpointVisible ³⁷²
- OnCMHintShow ³⁷²
- OnControlAction ³⁷³
- OnCopy ³⁷⁴
- OnGetItemCursor ³⁷⁴
- OnHTMLSaveImage ³⁷⁵
- OnImportFile ³⁷⁸
- OnImportPicture ³⁷⁸
- OnItemAction ³⁸⁰
- OnItemHint ³⁸¹
- OnJump ³⁸²
- OnLoadCustomFormat ³⁸³
- OnLoadDocument ³⁸⁴
- OnNewDocument ³⁸⁴
- OnPaint ³⁸⁴
- OnProgress ³⁸⁵
- OnReadField ³⁸⁶
- OnReadHyperlink ³⁸⁸
- OnReadMergeField ³⁹⁰
- OnRVDbClick ³⁹¹
- OnRVFControlNeeded ³⁹²
- OnRVFImageListNeeded ³⁹³
- OnRVFPictureNeeded ³⁹³
- OnRVMouseDown ³⁹⁴
- OnRVMouseMove ³⁹⁵
- OnRVMouseUp ³⁹⁶
- OnRVRightClick ³⁹⁷
- OnSaveComponentToFile ³⁹⁷
- OnSaveDocXExtra ³⁹⁹
- OnSaveHTMLExtra ⁴⁰¹
- OnSaveImage2 ⁴⁰²
- OnSaveItemToFile ⁴⁰⁴
- OnSaveParaToHTML ⁴⁰⁶
- OnSaveRTFExtra ⁴⁰⁶
- OnSelect ⁴⁰⁸
- OnSpellingCheck ⁴⁰⁹
- OnStyleTemplatesChange ⁴¹⁰
- OnTextFound ⁴¹⁰
- OnWriteHyperlink ⁴¹¹
- OnWriteObjectProperties ⁴¹³

Derived from TRVScroller⁽⁸¹³⁾

- OnHScrolled⁽⁸²¹⁾
- OnVScrolled⁽⁸²¹⁾

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(461))
```

DBRichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style⁽²⁵³⁾ property.

The most important property settings can be done in the component editor⁽²¹⁸⁾ 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⁽⁶¹¹⁾ to *True*.

If the application doesn't require the data-aware capabilities of TDBRichViewEdit, use TRichViewEdit⁽⁴⁶¹⁾ instead, to conserve system resources.

If the application doesn't require the editing capabilities of TDBRichViewEdit, use TDBRichView⁽⁵⁹⁴⁾ instead, to conserve system resources.

In Delphi XE2 or newer, there is another solution for data awareness: LiveBindings (see TRichView.Document⁽²³²⁾ property).

Hierarchy

TObject

TPersistent

TComponent

TControl

TWinControl

TCustomControl

TRVScroller⁽⁸¹³⁾

TCustomRichView⁽²¹⁰⁾

TCustomRichViewEdit⁽⁴⁶¹⁾

Field Type

This component must be linked to BLOB field.

If you use RVF⁽¹²⁴⁾ (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⁽⁶¹³⁾.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⁽²⁴⁷⁾ and FieldFormat⁽⁶¹³⁾=*rvdbRVF*, the component clears BackgroundBitmap⁽²²²⁾ before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF⁽¹²⁴⁾ (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⁽³⁸³⁾ 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⁽⁶¹³⁾ = *rvdbMarkdown*) or AllowMarkdown⁽⁶¹¹⁾, 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⁽⁵⁸⁵⁾ event allows saving data in your own format. If this event is not processed, documents are saved in FieldFormat⁽⁶¹³⁾.

Events

OnNewDocument⁽³⁸⁴⁾ 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⁽³⁸⁴⁾ occurs.

OnLoadCustomFormat⁽³⁸³⁾ and OnSaveCustomFormat⁽⁵⁸⁵⁾ allow saving and loading documents in custom formats.

Examples

Example 1: Using viewer-style methods in DBRichViewEdit⁽⁶²⁴⁾ (such as LoadRTF⁽³²⁶⁾)

Example 2: Inserting controls what can modify themselves in DBRichViewEdit⁽⁶²⁵⁾ (for example, inserting TEdit; user can modify its text)

6.1.4.1 Properties

In TDBRichViewEdit

- AllowMarkdown⁽⁶¹¹⁾
- AutoDeleteUnusedStyles⁽⁶¹¹⁾
- AutoDisplay⁽⁶¹²⁾
- CodePage⁽⁶¹²⁾
- DataField⁽⁶¹³⁾
- DataSource⁽⁶¹³⁾
- ▶ Field⁽⁶¹³⁾
- FieldFormat⁽⁶¹³⁾
- IgnoreEscape⁽⁶¹⁴⁾
- ReadOnly⁽⁶¹⁴⁾

Derived from TCustomRichViewEdit⁽⁴⁶¹⁾

- AcceptDragDropFormats⁽⁴⁷⁰⁾
- AcceptPasteFormats⁽⁴⁷⁰⁾
- ▶ ActualCurTextStyleNo⁽⁴⁷²⁾
- ▶ CurlItemNo⁽⁴⁷²⁾
- ▶ CurlItemStyle⁽⁴⁷³⁾
- CurParaStyleNo⁽⁴⁷³⁾
- CurTextStyleNo⁽⁴⁷⁴⁾

- CustomCaretInterval ⁴⁷⁴
- DefaultPictureVAlign ⁴⁷⁵
- EditorOptions ⁴⁷⁵
- ForceFieldHighlight ⁴⁷⁸
- LiveSpellingMode ⁴⁷⁹
- Modified ⁴⁷⁹
- ▶ OffsetInCurlItem ⁴⁷⁹
- ReadOnly ⁴⁸⁰
- SmartPopupProperties ⁴⁸⁰
- SmartPopupVisible ⁴⁸¹
- ▶ TopLevelEditor ⁴⁸¹
- UndoLimit ⁴⁸¹

Derived from TCustomRichView ²¹⁰

- AllowSelection ²²² (deprecated)
- AnimationMode ²²²
- BackgroundBitmap ²²²
- BackgroundStyle ²²³
- BiDiMode ²²⁴
- BottomMargin ²²⁵
- ClearLeft ²²⁵
- ClearRight ²²⁶
- Color ²²⁶
- CPEventKind ²²⁷
- Cursor ²²⁷
- DarkMode ²²⁸
- Delimiters ²²⁸
- DocObjects ²²⁹
- DocParameters ²³⁰
- DocProperties ²³¹
- ▶ DocumentHeight ²³³
- DocumentPixelsPerInch ²³³
- DolnPaletteMode ²³⁴
- ▶ FirstItemVisible ²³⁶
- FirstJumpNo ²³⁶
- HTMLReadProperties ²³⁷
- HTMLSaveProperties ²³⁷
- ▶ ItemCount ²³⁷
- ▶ LastItemVisible ²³⁸
- LeftMargin ²³⁸
- MarkdownProperties ²³⁹
- MaxLength ²³⁹
- MaxTextWidth ²³⁹
- MinTextWidth ²⁴⁰
- NoteText ²⁴⁰
- Options ²⁴⁰
- PageBreaksBeforeItems ²⁴⁴
- RightMargin ²⁴⁴

- RTFOptions ⁽²⁴⁵⁾
- RTFReadProperties ⁽²⁴⁶⁾
- ▶ RVData ⁽²⁴⁷⁾
- RVFOptions ⁽²⁴⁷⁾
- RVFParaStylesReadMode ⁽²⁵¹⁾
- RVFTextStylesReadMode ⁽²⁵¹⁾
- ▶ RVFWarnings ⁽²⁵¹⁾
- SingleClick ⁽²⁵³⁾ (deprecated)
- SelectionHandlesVisible ⁽²⁵³⁾ (D2010+)
- **Style** ⁽²⁵³⁾
- StyleTemplateInsertMode ⁽²⁵⁴⁾
- TabNavigation ⁽²⁵⁵⁾
- TopMargin ⁽²⁵⁵⁾
- UseStyleTemplates ⁽²⁵⁶⁾
- UseVCLThemes ⁽²⁵⁷⁾
- VAlign ⁽²⁵⁷⁾
- VSmallStep ⁽²⁵⁷⁾
- WordWrap ⁽²⁵⁸⁾

Derived from TRVScroller ⁽⁸¹³⁾

- BorderStyle ⁽⁸¹⁶⁾
- ▶ HScrollMax ⁽⁸¹⁶⁾
- HScrollPos ⁽⁸¹⁶⁾
- HScrollVisible ⁽⁸¹⁶⁾
- ▶ InplaceEditor ⁽⁸¹⁷⁾
- NoVScroll ⁽⁸¹⁷⁾
- Tracking ⁽⁸¹⁸⁾
- UseXPThemes ⁽⁸¹⁸⁾
- ▶ VScrollMax ⁽⁸¹⁸⁾
- VScrollPos ⁽⁸¹⁹⁾
- VScrollVisible ⁽⁸¹⁹⁾
- WheelStep ⁽⁸¹⁹⁾

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 (FieldFormat⁶¹³ = *rvdbMarkdown*)
- as a plain text, otherwise

Default value:

False

See also

- CodePage⁶¹²
- TRichViewEdit⁴⁶¹.Document⁴²².AllowMarkdown⁴²² (for LiveBindings)

6.1.4.1.2 TDBRichViewEdit.AutoDeleteUnusedStyles

If this property is set to *True*, the editor deletes unused text⁶⁵⁴, paragraph⁶⁴⁸ and list⁶⁴⁶ styles before posting to the database field.

property AutoDeleteUnusedStyles: Boolean;

(introduced in version 1.7)

If FieldFormat⁶¹³ = *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 editor must not share TRVStyle⁶³⁰ component with other TRichView controls (others TRichView controls must not be linked to the same TRVStyle component via Style²⁵³ property) and
- collections of text, paragraph and list styles must be saved together with documents in the RVF field (see RVFOptions²⁴⁷, *rvfoSaveTextStyles* and *rvfoSaveParaStyles* must be included for **AutoDeleteUnusedStyles** = *True*).

Default value:

False

See also

- TRichView⁽²¹⁰⁾.Document⁽²³²⁾.AutoDeleteUnusedStyles⁽⁴²³⁾ (for LiveBindings)
- TDBRichView⁽⁵⁹⁴⁾.AutoDeleteUnusedStyles⁽⁵⁹⁹⁾

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⁽⁶²²⁾ 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(996);
```

(introduced in version 23)

The default value of this property is *RV_CP_DEFAULT*⁽¹⁰⁵⁰⁾. 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⁽¹⁰⁵⁰⁾

See also

- FieldFormat⁽⁶¹³⁾
- TRichViewEdit⁽⁴⁶¹⁾.Document⁽⁴²²⁾.CodePage⁽⁴²³⁾ (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⁽⁶¹³⁾ 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⁽⁶¹³⁾ 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⁽¹²⁴⁾ (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⁽⁶¹³⁾

6.1.4.1.8 TDBRichViewEdit.FieldFormat

Format for saving data in the database field.

property FieldFormat: TRVDBFieldFormat⁽⁹⁹⁸⁾;

(introduced in version 1.6)

This property allows changing the field saving format to RVF (RichView Format)⁽¹²⁴⁾, 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⁽⁵⁸⁵⁾ 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`⁶¹¹).

Default value:

`rvdbRVF`

See also

- `TRichViewEdit`⁴⁶¹.`Document`²³².`FieldFormat`⁴²⁴ (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`⁴⁶¹.`Document`⁴²².`IgnoreEscape`⁴²⁴ (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`⁴⁸⁰ property is overridden in `TDBRichViewEdit` to get/set read-only mode in the linked dataset.

6.1.4.2 Methods

In `TDBRichViewEdit`

`Change`⁶²¹

`Create`⁶²¹

`Destroy`⁶²¹

`LoadField`⁶²²

Derived from `TCustomRichViewEdit`

`AdjustControlPlacement`⁴⁸⁹

`AdjustControlPlacement2`⁴⁸⁹

-  [ApplyListStyle](#) ⁽⁴⁹⁰⁾
-  [ApplyParaStyle](#) ⁽⁴⁹⁰⁾
-  [ApplyParaStyleConversion](#) ⁽⁴⁹¹⁾
-  [ApplyParaStyleTemplate](#) ⁽⁴⁹¹⁾
-  [ApplyStyleConversion](#) ⁽⁴⁹²⁾
-  [ApplyStyleTemplate](#) ⁽⁴⁹³⁾
-  [ApplyTextStyle](#) ⁽⁴⁹⁴⁾
-  [ApplyTextStyleTemplate](#) ⁽⁴⁹⁴⁾
- [BeginItemModify](#) ⁽⁴⁹⁵⁾
- [BeginUndoCustomGroup](#) ⁽⁴⁹⁶⁾
- [BeginUndoCustomGroup2](#) ⁽⁴⁹⁶⁾
- [BeginUndoGroup](#) ⁽⁴⁹⁷⁾
- [BeginUndoGroup2](#) ⁽⁴⁹⁷⁾
- [BeginUpdate](#) ⁽⁴⁹⁸⁾
- [CanChange](#) ⁽⁴⁹⁸⁾
- [CanPaste](#) ⁽⁴⁹⁸⁾
- [CanPasteHTML](#) ⁽⁴⁹⁹⁾
- [CanPasteRTF](#) ⁽⁴⁹⁹⁾
- [CanPasteRVF](#) ⁽⁴⁹⁹⁾
- [Change](#) ⁽⁴⁹⁹⁾
-  [ChangeListLevels](#) ⁽⁵⁰⁰⁾
-  [ChangeStyleTemplates](#) ⁽⁵⁰⁰⁾
- [Clear](#) ⁽⁵⁰¹⁾
-  [ClearTextFlow](#) ⁽⁵⁰¹⁾
- [ClearUndo](#) ⁽⁵⁰¹⁾
-  [ConvertToHotPicture](#) ⁽⁵⁰¹⁾
-  [ConvertToPicture](#) ⁽⁵⁰²⁾
- [Create](#) ⁽⁵⁰²⁾
-  [CutDef](#) ⁽⁵⁰²⁾
-  [DeleteSelection](#) ⁽⁵⁰³⁾
- [Destroy](#) ⁽⁵⁰³⁾
- [EndItemModify](#) ⁽⁵⁰³⁾
- [EndUndoGroup2](#) ⁽⁴⁹⁷⁾
- [EndUndoCustomGroup2](#) ⁽⁴⁹⁶⁾
- [EndUpdate](#) ⁽⁵⁰³⁾
- [GetCheckpointAtCaret](#) ⁽⁵⁰³⁾
- [GetCurrentBreakInfo](#) ⁽⁵⁰⁴⁾
- [GetCurrentBulletInfo](#) ⁽⁵⁰⁵⁾
- [GetCurrentCheckpoint](#) ⁽⁵⁰⁶⁾
- [GetCurrentControllInfo](#) ⁽⁵⁰⁶⁾
- [GetCurrentHotspotInfo](#) ⁽⁵⁰⁷⁾
- [GetCurrentItemExtraIntProperty](#) ⁽⁵¹⁰⁾
- [GetCurrentItemExtraIntPropertyEx](#) ⁽⁵¹⁰⁾
- [GetCurrentItemExtraStrProperty](#) ⁽⁵¹⁰⁾
- [GetCurrentItemExtraStrPropertyEx](#) ⁽⁵¹⁰⁾
- [GetCurrentItem](#) ⁽⁵⁰⁸⁾
- [GetCurrentItemEx](#) ⁽⁵⁰⁹⁾

- GetCurrentItemText -A -W ⁵¹¹
- GetCurrentItemVAlign ⁵¹²
- GetCurrentLineCol ⁵¹²
- GetCurrentMisspelling ⁵¹³
- GetCurrentPictureInfo ⁵¹³
- GetCurrentTag ⁵¹⁴
- GetCurrentTextInfo ⁵¹⁴
-  InsertBreak ⁵¹⁵
-  InsertBullet ⁵¹⁵
-  InsertCheckpoint ⁵¹⁶
-  InsertControl ⁵¹⁷
-  InsertDocXFromFileEd ⁵¹⁸ (D2009+ or with Delphi ZLib)
-  InsertDocXFromStreamEd ⁵¹⁸ (D2009+ or with Delphi ZLib)
-  InsertHotPicture ⁵¹⁹
-  InsertHotspot ⁵²⁰
-  InsertHTMLFromFileEd ⁵²¹
-  InsertHTMLFromStreamEd ⁵²¹
-  InsertItem ⁵²²
-  InsertMarkdownFromFileEd ⁵²³
-  InsertMarkdownFromStreamEd ⁵²⁴
-  InsertOEMTextFromFile ⁵²⁵
-  InsertPageBreak ⁵²⁵
-  InsertPicture ⁵²⁶
-  InsertRTFFFromFileEd ⁵²⁷
-  InsertRTFFFromStreamEd ⁵²⁸
-  InsertRVFFFromFileEd ⁵²⁸
-  InsertRVFFFromStreamEd ⁵²⁹
-  InsertStringTag -A -W ⁵³⁰
-  InsertTab ⁵³¹
-  InsertText -A -W ⁵³²
-  InsertTextFromFile -W ⁵³³
- MoveCaret ⁵³⁴
-  Paste ⁵³⁴
-  PasteBitmap ⁵³⁵
-  PasteHTML ⁵³⁶
-  PasteGraphicFile ⁵³⁵
-  PasteMetafile ⁵³⁷
-  PasteRTF ⁵³⁷
-  PasteRVF ⁵³⁸
-  PasteText -A -W ⁵³⁸
-  PasteURL ⁵³⁹
-  Redo ⁵⁴⁰
- RedoAction ⁵⁴⁰
- RedoName ⁵⁴⁰
-  RemoveCheckpointAtCaret ⁵⁴¹
-  RemoveCheckpointEd ⁵⁴¹

-  RemoveCurrentCheckpoint ⁵⁴¹
-  RemoveCurrentPageBreak ⁵⁴²
-  RemoveLists ⁵⁴²
-  ResizeControl ⁵⁴²
-  ResizeCurrentControl ⁵⁴³
- SearchText -A -W ⁵⁴³
- SelectCurrentLine ⁵⁴⁵
- SelectCurrentWord ⁵⁴⁵
-  SetBackgroundImageEd ⁵⁴⁶
-  SetBreakInfoEd ⁵⁴⁷
-  SetBulletInfoEd ⁵⁴⁸
-  SetCheckpointInfoEd ⁵⁴⁹
-  SetControlInfoEd ⁵⁵⁰
-  SetCurrentBreakInfo ⁵⁵¹
-  SetCurrentBulletInfo ⁵⁵²
-  SetCurrentCheckpointInfo ⁵⁵³
-  SetCurrentControlInfo ⁵⁵⁴
- SetFooter ³⁵⁵
- SetHeader ³⁵⁶
-  SetCurrentHotspotInfo ⁵⁵⁵
-  SetCurrentItemExtraIntProperty ⁵⁵⁶
-  SetCurrentItemExtraIntPropertyEx ⁵⁵⁶
-  SetCurrentItemExtraStrProperty ⁵⁵⁷
-  SetCurrentItemExtraStrPropertyEx ⁵⁵⁷
-  SetCurrentItemText -A -W ⁵⁵⁷
-  SetCurrentItemVAlign ⁵⁵⁸
-  SetCurrentPictureInfo ⁵⁵⁹
-  SetCurrentTag ⁵⁶⁰
-  SetFloatPropertyEd ⁵⁴⁵
-  SetHotspotInfoEd ⁵⁶⁰
-  SetIntPropertyEd ⁵⁴⁵
-  SetItemExtraIntPropertyEd ⁵⁶²
-  SetItemExtraIntPropertyExEd ⁵⁶²
-  SetItemExtraStrPropertyEd ⁵⁶²
-  SetItemExtraStrPropertyExEd ⁵⁶²
-  SetItemTagEd ⁵⁶³
-  SetItemTextEd -A -W ⁵⁶⁴
-  SetItemVAlignEd ⁵⁶⁵
-  SetPictureInfoEd ⁵⁶⁵
-  SetStrPropertyEd ⁵⁴⁵
- SetUndoGroupMode ⁵⁶⁷
-  Undo ⁵⁶⁷
- UndoAction ⁵⁶⁷
- UndoName ⁵⁶⁸

Derived from TCustomRichView ²¹⁰

-  AddBreak ²⁶²
-  AddBullet ²⁶³
-  AddCheckpoint ²⁶⁴
-  AddControl ²⁶⁵
-  AddFmt ²⁶⁶
-  AddHotPicture ²⁶⁷
-  AddHotspot ²⁶⁸
-  AddItem ²⁶⁹
-  AddNL -A -W ²⁷⁰
-  AddPicture ²⁷²
-  AddTab ²⁷³
-  AddTextNL -A -W ²⁷⁴
-  AppendFrom ²⁷⁵
-  AppendRVFFromStream ²⁷⁶
- AssignSoftPageBreaks ²⁷⁷
- BeginOleDrag ²⁷⁸
-  Clear ²⁷⁹
- ClearLiveSpellingResults ²⁷⁹
- ClearSoftPageBreaks ²⁷⁹
- ClientToDocument, DocumentToClient ²⁸⁰
- ConvertDocToPixels, ConvertDocToTwips, ConvertDocToEMU, ConvertDocToDifferentUnits ²⁸⁰
- Copy ²⁸¹
- CopyDef ²⁸¹
- CopyImage ²⁸¹
- CopyRTF ²⁸²
- CopyRVF ²⁸²
- CopyText -A -W ²⁸³
- Create ²⁸³
-  DeleteItems ²⁸³
- DeleteMarkedStyles ²⁸⁴
- DeleteParas ²⁸⁵
-  DeleteSection ²⁸⁵
- DeleteUnusedStyles ²⁸⁶
- Deselect ²⁸⁶
- Destroy ²⁸⁶
- FindCheckPointByName ²⁸⁷
- FindCheckPointByTag ²⁸⁷
- FindControllItemNo ²⁸⁷
- Format ²⁸⁸
- FormatTail ²⁸⁸
- GetBreakInfo ²⁸⁹
- GetBulletInfo ²⁸⁹
- GetCheckpointByNo ²⁹¹
- GetCheckpointInfo ²⁹¹
- GetCheckpointItemNo ²⁹¹

GetCheckpointNo²⁹²
GetCheckpointXY²⁹²
GetCheckpointY²⁹²
GetCheckpointYEx²⁹³
GetControllInfo²⁹³
GetFirstCheckpoint²⁹⁴
GetFocusedItem²⁹⁵
GetHotspotInfo²⁹⁵
GetItem²⁹⁶
GetItemAt²⁹⁷
GetItemCheckpoint²⁹⁸
GetItemCoords²⁹⁸
GetItemCoordsEx²⁹⁹
GetItemExtraIntProperty³⁰⁰
GetItemExtraIntPropertyEx³⁰⁰
GetItemExtraStrProperty³⁰⁰
GetItemExtraStrPropertyEx³⁰⁰
GetItemNo³⁰¹
GetItemPara³⁰¹
GetItemStyle³⁰²
GetItemTag³⁰²
GetItemText -A -W³⁰³
GetItemVAlign³⁰³
GetJumpPointLocation³⁰⁴
GetJumpPointY³⁰⁵
GetLastCheckpoint³⁰⁵
GetLineNo³⁰⁶
GetListMarkerInfo³⁰⁷
GetNextCheckpoint³⁰⁷
GetOffsAfterItem³⁰⁸
GetOffsBeforeItem³⁰⁹
GetPictureInfo³¹⁰
GetPrevCheckpoint³¹¹
GetRealDocumentPixelsPerInch³¹¹
GetSelectedImage³¹²
GetSelectionBounds³¹²
GetSelText -W³¹³
GetTextInfo³¹³
GetWordAt -A -W³¹⁴
 InsertRVFFromStream³¹⁶
IsFromNewLine³¹⁷
IsParaStart³¹⁸
LiveSpellingValidateWord³¹⁸
 LoadDocX³¹⁸ (D2009+ or with Delphi ZLib)
 LoadDocXFromStream³¹⁹ (D2009+ or with Delphi ZLib)
 LoadFromFile³²⁰
 LoadFromFileEx³²⁰

-  LoadFromStream ⁽³²¹⁾
-  LoadFromStreamEx ⁽³²¹⁾
-  LoadHTML ⁽³²²⁾
-  LoadHTMLFromStream ⁽³²⁴⁾
-  LoadMarkdown ⁽³²⁵⁾
-  LoadMarkdownFromStream ⁽³²⁵⁾
-  LoadRTF ⁽³²⁶⁾
-  LoadRTFFromStream ⁽³²⁷⁾
-  LoadRVF ⁽³²⁸⁾
-  LoadRVFFromStream ⁽³²⁹⁾
-  LoadText -W ⁽³³⁰⁾
-  LoadTextFromStream -W ⁽³³¹⁾
- MarkStylesInUse ⁽³³¹⁾
- RefreshListMarkers ⁽³³²⁾
- Reformat ⁽³³²⁾
-  RemoveCheckpoint ⁽³³²⁾
- ResetAnimation ⁽³³³⁾
- SaveDocX ⁽³³³⁾
- SaveDocXToStream ⁽³³⁴⁾
- SaveHTML ⁽³³⁵⁾
- SaveHTMLToStream ⁽³³⁸⁾
- SavePicture ⁽³⁴²⁾
- SaveRTF ⁽³⁴³⁾
- SaveRTFToStream ⁽³⁴⁴⁾
- SaveRVF ⁽³⁴⁴⁾
- SaveRVFToStream ⁽³⁴⁴⁾
- SaveText -W ⁽³⁴⁵⁾
- SaveTextToStream -W ⁽³⁴⁵⁾
- SearchText -A -W ⁽³⁴⁶⁾
- SelectAll ⁽³⁴⁸⁾
- SelectControl ⁽³⁴⁸⁾
- SelectionExists ⁽³⁴⁹⁾
- SelectWordAt ⁽³⁴⁹⁾
- SetAddParagraphMode ⁽³⁵⁰⁾
-  SetBreakInfo ⁽³⁵¹⁾
-  SetBulletInfo ⁽³⁵²⁾
-  SetCheckpointInfo ⁽³⁵³⁾
-  SetControlInfo ⁽³⁵⁴⁾
-  SetHotspotInfo ⁽³⁵⁷⁾
-  SetItemExtraIntProperty ⁽³⁵⁸⁾
-  SetItemExtraIntPropertyEx ⁽³⁵⁸⁾
-  SetItemExtraStrProperty ⁽³⁵⁹⁾
-  SetItemExtraStrPropertyEx ⁽³⁵⁹⁾
-  SetItemTag ⁽³⁶⁰⁾
-  SetItemText -A -W ⁽³⁶⁰⁾
-  SetItemVAlign ⁽³⁶¹⁾

-  [SetListMarkerInfo](#) ⁽³⁶²⁾
-  [SetPictureInfo](#) ⁽³⁶³⁾
- [SetSelectionBounds](#) ⁽³⁶⁵⁾
- [StartAnimation](#) ⁽³⁶⁶⁾
- [StartLiveSpelling](#) ⁽³⁶⁷⁾
- [StopAnimation](#) ⁽³⁶⁷⁾
- [StoreSearchResult](#) ⁽³⁶⁷⁾
- [UpdatePaletteInfo](#) ⁽³⁶⁸⁾

Derived from TRVScroller ⁽⁸¹³⁾

[ScrollTo](#) ⁽⁸²⁰⁾

6.1.4.2.1 TDBRichViewEdit.Change

Generates [OnChange](#) ⁽⁵⁷²⁾ event.

```
procedure Change; override;
```

[TCustomRichViewEdit.Change](#) ⁽⁴⁹⁹⁾ 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](#) ⁽⁶²⁴⁾.

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 ⁽²¹⁸⁾ (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](#) ⁽⁶²¹⁾ 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⁽⁶¹²⁾ 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⁽⁵⁸⁵⁾

Derived from TCustomRichViewEdit

- OnAfterOleDrop⁽⁵⁷⁰⁾
- OnBeforeOleDrop⁽⁵⁷⁰⁾
- OnCaretGetOut⁽⁵⁷¹⁾
- OnCaretMove⁽⁵⁷²⁾
- OnChange⁽⁵⁷²⁾
- OnChanging⁽⁵⁷²⁾
- OnCheckStickingItems⁽⁵⁷³⁾
- OnCurParaStyleChanged⁽⁵⁷³⁾
- OnCurTextStyleChanged⁽⁵⁷⁴⁾
- OnDrawCurrentLine⁽⁵⁷⁴⁾
- OnDrawCustomCaret⁽⁵⁷⁵⁾
- OnDropFile⁽⁵⁷⁶⁾
- OnDropFiles⁽⁵⁷⁶⁾
- OnItemTextEdit⁽⁵⁷⁸⁾
- OnMeasureCustomCaret⁽⁵⁷⁸⁾
- OnOleDragEnter⁽⁵⁷⁹⁾
- OnOleDragLeave⁽⁵⁷⁹⁾
- OnOleDragOver⁽⁵⁸⁰⁾
- OnOleDrop⁽⁵⁸¹⁾
- OnParaStyleConversion⁽⁵⁸²⁾
- OnPaste⁽⁵⁸⁴⁾
- OnSaveCustomFormat⁽⁵⁸⁵⁾
- OnSmartPopupClick⁽⁵⁸⁵⁾
- OnStyleConversion⁽⁵⁸⁵⁾

Derived from TCustomRichView⁽²¹⁰⁾

- OnAddStyle⁽³⁷⁰⁾
- OnAfterDrawImage⁽³⁷⁰⁾
- OnAssignImageFileName⁽³⁷¹⁾
- OnCheckpointVisible⁽³⁷²⁾ (doesn't work in editor)

- OnCMHintShow ³⁷²
- OnControlAction ³⁷³
- OnCopy ³⁷⁴
- OnGetItemCursor ³⁷⁴
- OnHTMLSaveImage ³⁷⁵
- OnImportFile ³⁷⁸
- OnImportPicture ³⁷⁸
- OnItemAction ³⁸⁰
- OnItemHint ³⁸¹
- OnJump ³⁸²
- OnLoadCustomFormat ³⁸³
- OnLoadDocument ³⁸⁴
- OnNewDocument ³⁸⁴
- OnPaint ³⁸⁴
- OnProgress ³⁸⁵
- OnReadField ³⁸⁶
- OnReadHyperlink ³⁸⁸
- OnReadMergeField ³⁹⁰
- OnRVDbClick ³⁹¹
- OnRVFControlNeeded ³⁹²
- OnRVFImageListNeeded ³⁹³
- OnRVFPictureNeeded ³⁹³
- OnRVMouseDown ³⁹⁴
- OnRVMouseMove ³⁹⁵
- OnRVMouseUp ³⁹⁶
- OnRVRightClick ³⁹⁷
- OnSaveComponentToFile ³⁹⁷
- OnSaveDocXExtra ³⁹⁹
- OnSaveHTMLExtra ⁴⁰¹
- OnSaveImage2 ⁴⁰²
- OnSaveItemToFile ⁴⁰⁴
- OnSaveParaToHTML ⁴⁰⁶
- OnSaveRTFExtra ⁴⁰⁶
- OnSelect ⁴⁰⁸
- OnSpellingCheck ⁴⁰⁹
- OnStyleTemplatesChange ⁴¹⁰
- OnTextFound ⁴¹⁰
- OnWriteHyperlink ⁴¹¹
- OnWriteObjectProperties ⁴¹³

Derived from TRVScroller ⁸¹³

- OnHScrolled ⁸²¹
- OnVScrolled ⁸²¹

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 ⁶²⁴
- Inserting controls that can modify themselves in DBRichViewEdit ⁶²⁵

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 ²¹⁰).

DBRichViewEdit1 edits a field of table Table1. We want to load RTF file.

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenFileDialog1.Execute and DBRichViewEdit1.CanChange 498 then
    begin
      DBRichViewEdit1.Clear 501;
      DBRichViewEdit1.LoadRTF 326 (OpenDialog1.FileName);
      DBRichViewEdit1.Change 621;
      DBRichViewEdit1.Format 288;
    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 OpenFileDialog1.Execute then

```

```
DBRichViewEdit1.InsertRTFFFromFileEd527(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 TDBRichViewEdit⁶⁰⁶, 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.CanChange498 then
```

```
    DBRichViewEdit1.Change621
```

```
  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.InsertControl517(' ', Edit, rvvaBaseline);
```

```
end;
```

3) You'll need to assign this event to all edits loaded from the database field with RichView documents. Use OnControlAction³⁷³ 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 TRichView²¹⁰, TRichViewEdit⁴⁶¹, TDBRichView⁵⁹⁴, or TDBRichViewEdit⁶⁰⁶.

TRVPrintPreview control can display preview only if it is linked to RVPrint⁶²⁸, and RVPrint is ready (it has a document assigned and pages formatted)

Unit [VCL/FMX] RVPP / fmxRVPP.

Syntax

```
TRVPrintPreview = class(TCustomRVPrintPreview832)
```

Hierarchy

TObject

TPersistent

TComponent

TControl

TWinControl

TCustomControl

*TRVScroller*⁸¹³

*TCustomRVPrintPreview*⁸³²

Information

See more info in the topic about the ancestor of TRVPrintPreview, TCustomRVPrintPreview⁸³².

The main property of this component is RVPrint⁶²⁸, allowing to link it with TRVPrint⁷⁵⁹ object.

In the Lazarus for Linux version, it's highly recommended to assign CachePageImage⁶²⁸ = *True*.

6.1.5.1 Properties

In TRVPrintPreview

- CachePageImage⁶²⁸ [VCL,LCL]
- RVPrint⁶²⁸

Derived from TCustomRVPrintPreview⁸³²

- ClickMode⁸³⁶
- Color⁸³⁶ [VCL, LCL]
- DarkModeUI⁸³⁷
- Fill⁸³⁶ [FMX]
- MarginsPen⁸³⁷ [VCL,LCL]
- MarginsStroke⁸³⁷ [FMX]
- PageBorderColor⁸³⁷
- PageBorderWidth⁸³⁸
- PageNo⁸³⁸
- PrintableAreaPen⁸³⁸ [VCL,LCL]
- PrintableAreaStroke⁸³⁸ [FMX]
- ShadowColor⁸³⁹
- ShadowWidth⁸³⁹
- ZoomInCursor⁸³⁹
- ZoomMode⁸⁴⁰
- ZoomOutCursor⁸⁴⁰
- ZoomPercent⁸⁴¹

Derived from TRVScroller⁸¹³

- BorderStyle⁸¹⁶ [VCL]

- ▶ HScrollMax⁸¹⁶
- HScrollPos⁸¹⁶
- HScrollVisible⁸¹⁶ [VCL,LCL]
- ▶ InplaceEditor⁸¹⁷ (not used in this component)
- NoVScroll⁸¹⁷
- Tracking⁸¹⁸
- UseXPThemes⁸¹⁸ [VCL,LCL]
- ▶ VScrollMax⁸¹⁸
- VScrollPos⁸¹⁹
- VScrollVisible⁸¹⁹ [VCL,LCL]
- WheelStep⁸¹⁹

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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
- ShowScrollBar [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: TRVPrint759;
```

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 RVPrint⁶²⁸ 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 TCustomRVPrintPreview⁸³²

```
Create841
First841
Last842
Next842
Prev842
SetZoom843
```

Derived from TRVScroller⁸¹³

```
ScrollTo820 (not used in this component)
```

6.1.5.3 Events

Derived from TCustomRVPrintPreview ⁸³²

- OnZoomChanged ⁸⁴³

Derived from TRVScroller ⁸¹³

- OnHScrolled ⁸²¹
- OnVScrolled ⁸²¹

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 ²⁰⁹ | Additional Components

Icon	Class Name	Description
	TRVStyle ⁶³⁰	Component for defining visual appearance of TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶ controls. It contains some properties that can be used by several TRichView controls.
	TRVPrint ⁷⁵⁹	Component for printing documents contained in TRichView ²¹⁰ , TRichViewEdit ⁶⁰⁶ , TDBRichView ⁵⁹⁴ , TDBRichViewEdit ⁶⁰⁶ .
	TRVReportHelper ⁸⁰⁴	Component for drawing documents on different canvas (screen, printer, image, etc.)

Icon	Class Name	Description
	TRVOfficeConverter ⁽⁷⁹⁵⁾	Component allowing to use Microsoft Office text converters (import and export in different file formats) [VCL and LCL]
	TRVDataSourceLink ⁽⁸¹¹⁾	Component allowing assigning TDataSource component to controls inserted in TRichView ⁽²¹⁰⁾ , TRichViewEdit ⁽⁶⁰⁶⁾ , TDBRichView ⁽⁵⁹⁴⁾ , TDBRichViewEdit ⁽⁶⁰⁶⁾ , TSRichViewEdit, TDBSRichViewEdit controls. [VCL and LCL]
	TRVSpellChecker ⁽⁷⁸⁴⁾	Component for spelling checking

6.2.1 TRVStyle

This invisible at run-time component is used for defining a visual appearance of RichView⁽²¹⁰⁾ 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⁽²⁵³⁾ 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⁽²¹⁰⁾, RichViewEdit⁽⁴⁶¹⁾, DBRichView⁽⁵⁹⁴⁾, DBRichViewEdit⁽⁶⁰⁶⁾ controls.

But if you use the following:

- TCustomRichView.DeleteUnusedStyles⁽²⁸⁶⁾;
- TDBRichViewEdit.AutoDeleteUnusedStyles⁽⁶¹¹⁾=True;
- TCustomRichView.LoadRVF⁽³²⁸⁾, LoadRVFFromStream⁽³²⁷⁾, and RVFTextStylesReadMode⁽²⁵¹⁾ or RVFParaStylesReadMode⁽²⁵¹⁾ <> rvf_ignore;
- TDBRichView or TDBRichViewEdit, and RVFTextStylesReadMode⁽²⁵¹⁾ or RVFParaStylesReadMode⁽²⁵¹⁾ <> rvf_ignore

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⁽⁶⁵⁴⁾, ParaStyles⁽⁶⁴⁸⁾, or ListStyles⁽⁶⁴⁶⁾;

- adds “Convert Lengths to twips” (... to pixels, ... to EMU) item in the context menu, allowing to convert all RVStyle properties measured in Units⁽⁶⁵⁵⁾.
- adds 3 items into the context menu for component: “Edit Style Templates”, “Save Style Templates...”, “Load Style Templates...” which invoke property editors for StyleTemplates⁽⁶⁵²⁾, 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⁽⁶⁵⁴⁾ – collection of text attributes (styles), a collection of TFontInfo⁽⁷¹²⁾ items;
- ParaStyles⁽⁶⁴⁸⁾ – collection of paragraph attributes (styles), a collection of TParaInfo⁽⁷²⁷⁾ items;
- ListStyles⁽⁶⁴⁶⁾ – collection of paragraph list styles (bullets and numbering), a collection of TRVListInfo⁽⁷³¹⁾ items.

Style templates ("real" styles)

- StyleTemplates⁽⁶⁵²⁾ – collection of "real" styles, controlling properties of TextStyles⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾;

Units of measurement

- Units⁽⁶⁵⁵⁾ defines measure units for properties of this component and documents in linked TRichView controls.
- UnitsPixelsPerInch⁽⁶⁵⁵⁾ defines DPI for values measured in pixels.

RVStyle allows to set some colors for associated controls:

- Color⁽⁶³⁵⁾ – background color;
- SelColor, InactiveSelColor⁽⁶⁴¹⁾ – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor⁽⁶⁴¹⁾ – colors of selected text (with and without focus);
- DisabledFontColor⁽⁶³⁷⁾ – text color for disabled controls.
- HoverColor⁽⁶⁴¹⁾ – default color of hypertext links under the mouse pointer;
- CheckpointColor, CheckpointEvColor⁽⁶³⁴⁾ – colors of *checkpoints* (if visible);
- PageBreakColor⁽⁶⁴⁷⁾ – color of explicit page breaks (if visible);
- FloatingLineColor⁽⁶³⁸⁾ – color of placeholders for left- and right-aligned items (if visible).
- GridColor, GridReadOnlyColor⁽⁶⁴⁰⁾ – colors of table grid lines (if visible); GridStyle, GridReadOnlyStyle⁽⁶⁴⁰⁾ define their pen styles.

RVStyle allows to set cursors:

JumpCursor⁽⁶⁴³⁾ – hypertext cursor for *hotspots*⁽¹⁷²⁾ and *hot-pictures*⁽¹⁶⁶⁾ (and default cursor for text hypertext items);

LineSelectCursor⁽⁶⁴³⁾ – cursor for the left margin.

Tabs:

- SpacesInTab⁽⁶⁵²⁾ – 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⁽¹⁶²⁾ is used;
- DefTabWidth⁽⁶³⁶⁾ – default distance between tab stops.

Properties related to label items⁽¹⁷⁶⁾ and item types inherited from them:

- FieldHighlightColor⁽⁶³⁷⁾ – color for highlighting label items.
- FieldHighlightType⁽⁶³⁸⁾ specifies when to highlight label items.
- FootnoteNumbering⁽⁶³⁷⁾ – numbering type for footnotes.
- FootnotePageReset⁽⁶⁴⁰⁾ – numbering mode for footnotes.

- EndnoteNumbering⁶³⁷ – numbering type for endnotes.
- SidenoteNumbering⁶³⁷ – numbering type for sidenotes.

Properties related to selection:

- SelectionStyle⁶⁴⁹ – visual appearance of selection (draw for objects or lines);
- SelectionMode⁶⁴⁹ – modes of making selection;
- SelectionHandleKind⁶⁴⁸ – visual appearance of touch screen selection handles;
- SelColor, InactiveSelColor⁶⁴¹ – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor⁶⁴¹ – colors of selected text (with and without focus).
- SelOpacity⁶⁴¹ – selection fill opacity (opaque and semitransparent selection is drawn differently)

Other properties

- TextEngine⁶⁵⁴ specifies which API is used for drawing and measuring text;
- FontQuality⁶³⁹ – font quality and anti-aliasing;
- DefCodePage⁶³⁵ – default code page for internal ANSI <->Unicode conversion;
- LineWrapMode⁶⁴⁴ specifies a line wrapping algorithm;
- UseSound⁶⁵⁶.

Using main TRVStyle

If another TRVStyle component is assigned to MainRVStyle⁶⁴⁷ property, most properties of this TRVStyle simply provide access to properties of this main TRVStyle.

(For example, SelColor⁶⁴¹ returns MainRVStyle⁶⁴⁷.SelColor⁶⁴¹).

Initially, this feature was implemented to allow using a single StyleTemplates⁶⁵² collection in multiple TRVStyle components. But in the new version of TRVStyle all properties of MainRVStyle are shared.

The following properties are not shared:

- TextStyles⁶⁵⁴
- ParaStyles⁶⁴⁸
- ListStyles⁶⁴⁶
- all events

Methods

You can save and load all information in RVStyle in ini-file:

- SaveINI⁶⁶³ saves RVStyle properties in ini-file;
- LoadINI⁶⁶¹ loads RVStyle properties from ini-file;

or registry:

- SaveReg⁶⁶⁴ saves RVStyle properties in ini-file;
- LoadReg⁶⁶² loads RVStyle properties from ini-file.

You can export styles in style sheet (for saving HTML files):

- SaveCSS⁶⁶²;
- SaveCSSToStream⁶⁶³.

You can search for a style with the specified properties:

- FindTextStyle, FindParaStyle⁶⁶¹

Events

Custom drawing:

- OnDrawCheckpoint⁶⁶⁷ – drawing checkpoint;
- OnDrawPageBreak⁶⁶⁸ – drawing page break;
- OnDrawParaBack⁶⁷⁰ – drawing background of paragraph;
- OnDrawTextBack⁶⁷³ – drawing background of text;
- OnApplyStyle⁶⁶⁵ – applying text style to Canvas;
- OnApplyStyleColor⁶⁶⁶ – applying text style color to Canvas;
- OnDrawStyleText⁶⁷¹ – drawing text of the given style;
- OnStyleHoverSensitive⁶⁷⁴ – "should the component be repainted when user moves the mouse pointer over hyperlink of the given style?".

6.2.1.1 Properties

In TRVStyle

- CheckpointColor⁶³⁴
- CheckpointEvColor⁶³⁴
- Color⁶³⁵
- CurrentItemColor⁶³⁵
- DefCodePage⁶³⁵
- DefTabWidth⁶³⁶
- DisabledFontColor⁶³⁷
- EndnoteNumbering⁶³⁷
- FieldHighlightColor⁶³⁷
- FieldHighlightType⁶³⁸
- FloatingLineColor⁶³⁸
- FontQuality⁶³⁹ [VCL and LCL]
- FootnoteNumbering⁶³⁷
- FootnotePageReset⁶⁴⁰
- FullRedraw⁶⁴⁰ [VCL and LCL]
- GridColor⁶⁴⁰
- GridReadOnlyColor⁶⁴⁰
- GridStyle⁶⁴⁰
- GridGridReadOnlyStyle⁶⁴⁰
- HoverColor⁶⁴¹
- InactiveSelColor⁶⁴¹
- InactiveSelTextColor⁶⁴¹
- InvalidPicture⁶⁴³
- JumpCursor⁶⁴³
- LineSelectCursor⁶⁴³
- LineWrapMode⁶⁴⁴
- ListStyles⁶⁴⁶
- LiveSpellingColor⁶⁴⁶
- MainRVStyle⁶⁴⁷
- PageBreakColor⁶⁴⁷
- ParaStyles⁶⁴⁸

- SelColor⁽⁶⁴¹⁾
- SelectionHandleKind⁽⁶⁴⁸⁾ (D2010+)
- SelectionMode⁽⁶⁴⁹⁾
- SelectionStyle⁽⁶⁴⁹⁾
- SelOpacity⁽⁶⁴¹⁾
- SelTextColor⁽⁶⁴¹⁾
- SidenoteNumbering⁽⁶³⁷⁾
- SoftPageBreakColor⁽⁶⁵¹⁾
- SpacesInTab⁽⁶⁵²⁾
- StyleTemplates⁽⁶⁵²⁾
- TextBackgroundKind⁽⁶⁵³⁾
- TextEngine⁽⁶⁵⁴⁾ [VCL and LCL]
- TextStyles⁽⁶⁵⁴⁾
- Units⁽⁶⁵⁵⁾
- UnitsPixelsPerInch⁽⁶⁵⁵⁾
- UseSound⁽⁶⁵⁶⁾

6.2.1.1.1 TRVStyle.CheckpointColor, CheckpointColor

A color of dotted lines showing positions of *checkpoints* in document.

```
property CheckpointColor: TRVColor(996);
property CheckpointEvColor: TRVColor(996);
```

If TRichView shows its *checkpoints* (*rvoShowCheckpoints* in TRichView.Options⁽²⁴⁰⁾), 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⁽⁶⁶⁷⁾ event.

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, these properties provide access to the corresponding properties of MainRVStyle⁽⁶⁴⁷⁾.

Default value

CheckpointColor: rvclGreen⁽¹⁰³⁸⁾

CheckpointEvColor: rvclLime⁽¹⁰³⁸⁾

See also events:

- OnDrawCheckpoint⁽⁶⁶⁷⁾.

See also properties of TCustomRichView:

- Options⁽²⁴⁰⁾.

See also:

- *Checkpoints* overview⁽⁸⁷⁾.

6.2.1.1.2 TRVStyle.Color

A default background color for TRichView component linked to this TRVStyle component.

```
property Color: TRVColor996 ;
```

This property may be overridden with TRichView.Color²²⁶ property.

TRichView must be linked with this TRVStyle (see TRichView.Style²⁵³).

If MainRVStyle⁶⁴⁷ is assigned, this property provides access to the corresponding property of MainRVStyle⁶⁴⁷.

Default value for VCL and LCL:

c/Window

Default value for FireMonkey:

AlphaColorRec.Null (a transparent document background, so a styled component background is visible, unless DarkMode²²⁸).

See also properties of TCustomRichView:

- Style²⁵³;
- Color²²⁶;
- BackgroundStyle²²³;
- BackgroundBitmap²²².

6.2.1.1.3 TRVStyle.CurrentItemColor

A color for drawing frame around the current item in TRichViewEdit⁴⁶¹.

```
property CurrentItemColor: TRVColor996 ;
```

(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 *c/None* to disable this effect.

TRichViewEdit must be linked with this TRVStyle (see TRichView.Style²⁵³).

If MainRVStyle⁶⁴⁷ is assigned, this property provides access to the corresponding property of MainRVStyle⁶⁴⁷.

Default value:

*rvclNone*¹⁰³⁸

6.2.1.1.4 TRVStyle.DefCodePage

A default code page used inside TRichView for automatic conversions of ANSI strings to Unicode and vice versa

```
property DefCodePage: TRVCodePage996 ;
```

(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⁽⁷⁰⁰⁾ 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⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

0 (Windows default code page, *CP_ACP*)

See also:

- Unicode in RichView⁽¹³⁰⁾.

6.2.1.1.5 TRVStyle.DefTabWidth

A default distance between tab stops

```
property DefTabWidth: TRVStyleLength(1027);
```

(introduced in v1.9)

This value is measured Style⁽²⁵³⁾.Units⁽⁶⁵⁵⁾.

Tab stops are started from the LeftMargin⁽²³⁸⁾ (or from the RightMargin⁽²⁴⁴⁾ for right-to-left⁽¹³²⁾ paragraphs).

Note: tabulators⁽¹⁶²⁾ can be inserted from text/RTF files or keyboard only if value of SpacesInTab⁽⁶⁵²⁾ property is 0 (otherwise tabs are inserted as several space characters).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

48

See also properties:

- ParaStyles⁽⁶⁴⁸⁾.Tabs⁽⁷²⁶⁾.

See also properties:

- Units of measurement in TRichView⁽¹³⁹⁾

6.2.1.1.6 TRVStyle.DefUnicodeStyle

```
property DefUnicodeStyle: Integer;
```

Obsolete property. Not available in FireMonkey version.

Default value:

-1

6.2.1.1.7 TRVStyle.DisabledFontColor

Defines a font color to use in disabled TRichView⁽²¹⁰⁾ controls.

```
property DisabledFontColor: TRVColor(996) ;
```

(Introduced in version 13)

If this property = *rvclNone*⁽¹⁰³⁸⁾, a disabled control uses the same text color as enabled control.

If *MainRVStyle*⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of *MainRVStyle*⁽⁶⁴⁷⁾.

Default value:

rvclNone⁽¹⁰³⁸⁾

6.2.1.1.8 TRVStyle.EndnoteNumbering

A numbering type for endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾, and sidenotes⁽¹⁸¹⁾.

```
property EndnoteNumbering: TRVSeqType
```

```
property FootnoteNumbering: TRVSeqType
```

```
property SidenoteNumbering: TRVSeqType
```

(introduced in version 10 and 15)

See *TRVSeqType*⁽¹⁰²⁶⁾ for possible values.

If you changed value of these properties while the document in the linked⁽²⁵³⁾ TRichView⁽²¹⁰⁾ control is displayed, call *RichView.RefreshSequences*⁽³³²⁾ and *RichView.Format*⁽²⁸⁸⁾.

If *MainRVStyle*⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of *MainRVStyle*⁽⁶⁴⁷⁾.

Default values:

- *EndnoteNumbering*: *rvseqLowerRoman*
- *FootnoteNumbering*: *rvseqDecimal*
- *SidenoteNumbering*: *rvseqLowerAlpha*

See also:

- *FootnotePageReset*⁽⁶⁴⁰⁾.

6.2.1.1.9 TRVStyle.FieldHighlightColor

A color for highlighting labels⁽¹⁷⁶⁾, numbered sequences⁽¹⁷⁷⁾, endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾, references to parent footnotes and endnotes⁽¹⁸⁴⁾.

```
property FieldHighlightColor: TRVColor(996) ;
```

(introduced in version 10)

See *FieldHighlightType*⁽⁶³⁸⁾ for details.

If *MainRVStyle*⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of *MainRVStyle*⁽⁶⁴⁷⁾.

Default value:

rvclBtnFace⁽¹⁰³⁸⁾

6.2.1.1.10 TRVStyle.FieldHighlightType

Specifies when `FieldHighlightColor`⁽⁶³⁷⁾ must be used for highlighting labels⁽¹⁷⁶⁾, numbered sequences⁽¹⁷⁷⁾, endnotes⁽¹⁷⁸⁾, footnotes⁽¹⁸⁰⁾, references to parent footnotes and endnotes⁽¹⁸⁴⁾.

type

```
TRVFieldHighlightType =
    (rvfhNever, rvfhCurrent, rvfhAlways);
```

property `FieldHighlightType`;

(introduced in version 10)

Type	Meaning
<i>rvfhNever</i>	Items are never highlighted.
<i>rvfhCurrent</i>	Item at the position of the caret is highlighted. Either the editor must have the input focus, or its <code>ForceFieldHighlight</code> ⁽⁴⁷⁸⁾ property must be <i>True</i> .
<i>rvfhAlways</i>	All items are always highlighted.

If `MainRVStyle`⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁽⁶⁴⁷⁾.

Default value:

rvfhCurrent

6.2.1.1.11 TRVStyle.FloatingLineColor

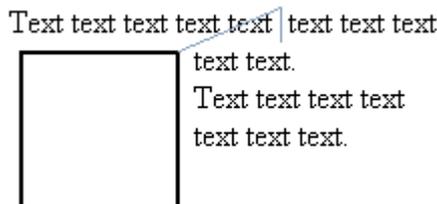
Color for displaying places of insertion for left- and right-aligned⁽¹⁰³³⁾ items, and markers for text box items⁽¹⁸³⁾.

property `FloatingLineColor`: `TRVColor`⁽⁹⁹⁶⁾;

(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`⁽⁴⁶¹⁾, 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`⁽²⁴⁰⁾, and `rvcFloatingLines` is included in `RVVisibleSpecialCharacters`⁽¹⁰⁶¹⁾

Assign `rvcNone`⁽¹⁰³⁸⁾ to prevent displaying these lines.

Text box items

A special mark is inserted in places of insertion of text box items ⁽¹⁸³⁾:

This is a line containing a text box. ¶

The marks are displayed in the following cases:

- in `TCustomRichViewEdit` ⁽⁴⁶¹⁾, 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` ⁽²⁴⁰⁾, and `rvcPlaceholders` is included in `RVVisibleSpecialCharacters` ⁽¹⁰⁶¹⁾.

If `FloatingLineColor=rvclNone` ⁽¹⁰³⁸⁾, `SpecialCharactersColor` ⁽⁶⁵²⁾ is used. If `SpecialCharactersColor=rvclNone` ⁽¹⁰³⁸⁾, `rvclBtnShadow` ⁽¹⁰³⁸⁾ is used.

 **ScaleRichView note:** in `TSRichViewEdit`, this color is also used to draw lines to places of insertion of sidenotes ⁽¹⁸¹⁾ and text box items, and around the edited floating box. Assign `clNone` to prevent displaying these lines.

If `MainRVStyle` ⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of `MainRVStyle` ⁽⁶⁴⁷⁾.

Default value (VCL and LCL / FireMonkey):

\$BDA28A / \$FF8AA2BD

6.2.1.1.12 TRVStyle.FontQuality

Specifies the font quality.

type

```
TFontQuality = (fqDefault, fqDraft, fqProof, fqNonAntialiased,
  fqAntialiased, fqClearType, fqClearTypeNatural);
```

property FontQuality: TFontQuality;

(introduced in version 15)

Type	Meaning
<i>fqDefault</i>	The quality is determined by the system settings.
<i>fqDraft</i>	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.
<i>fqProof</i>	The quality of the characters is important, so for the raster fonts scaling is disabled and the font closest in size is chosen.
<i>fqNonAntialiased</i>	Fonts are never antialiased.
<i>fqAntialiased</i>	Fonts are always antialiased if they support it. The size of the font cannot be too small or too large.
<i>fqClearType</i>	Fonts are rendered using the ClearType antialiasing method.

Type	Meaning
<i>fqClearTypeNatural</i>	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 MainRVStyle⁶⁴⁷ is assigned, this property provides access to the corresponding property of MainRVStyle⁶⁴⁷.

Default value:

fqDefault

6.2.1.1.13 TRVStyle.FootnotePageReset

Specifies whether to reset numbering of footnotes¹⁸⁰ 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.RTFReadProperties²⁴⁶.IgnoreNotes⁴⁵⁵=*False*.

If MainRVStyle⁶⁴⁷ is assigned, this property provides access to the corresponding property of MainRVStyle⁶⁴⁷.

Default value:

True

See also:

- FootnoteNumbering⁶³⁷

6.2.1.1.14 TRVStyle.FullRedraw

```
property FullRedraw: Boolean;
```

Obsolete property. Not available in FireMonkey version.

Default value:

False

6.2.1.1.15 TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle

Colors and styles of table¹⁷³ grid lines.

```
property GridColor: TRVColor996;
```

```
property GridReadOnlyColor: TRVColor996;
```

```
property GridStyle: TRVPenStyle1018;
```

```
property GridReadOnlyStyle: TRVPenStyle1018;
```

(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⁽²¹⁰⁾ or in TRichViewEdit⁽⁴⁶¹⁾ in read-only⁽⁴⁸⁰⁾ mode.

Table borders can be invisible in the following cases:

- their width is 0 (see BorderWidth⁽⁸⁵³⁾ and CellBorderWidth⁽⁸⁵⁵⁾ properties);
- some sides are hidden (see VisibleBorders⁽⁸⁶⁵⁾ and Cell⁽⁸⁹⁵⁾.VisibleBorders⁽⁹⁰⁶⁾ 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⁽²⁴⁰⁾.

You can hide/show table grid for the specific table, see *rvtoHideGridLines* in Table.Options⁽⁸⁶¹⁾ property.

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, these properties provide access to the corresponding properties of MainRVStyle⁽⁶⁴⁷⁾.

Default values:

- GridColor: *rvclBtnFace*⁽¹⁰³⁸⁾;
- GridReadOnlyColor: *rvclBtnFace*⁽¹⁰³⁸⁾;
- GridStyle: *psDot (VCL and LCL) / rvpsSolid*⁽¹⁰⁴²⁾ (*FireMonkey*)
- GridReadOnlyStyle: *rvpsClear*⁽¹⁰⁴²⁾.

6.2.1.1.16 TRVStyle HoverColor

Default a text hover color.

```
property HoverColor: TRVColor(996);
```

A text hover color is a text color under the mouse pointer.

All hypertext styles have their own hover color (see TFontInfo.HoverColor⁽⁷⁰²⁾). If they are equal to *rvclNone*⁽¹⁰³⁸⁾, this **HoverColor** is used.

Set to *rvclNone*⁽¹⁰³⁸⁾ to disable hover color effect.

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

rvclNone⁽¹⁰³⁸⁾

6.2.1.1.17 TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor, SelOpacity

Color of text and background of selected text in TRichView components.

```
property SelColor: TRVColor(996);  
property SelTextColor: TRVColor(996);  
property InactiveSelColor: TRVColor(996);  
property InactiveSelTextColor: TRVColor(996);  
property SelOpacity: TRVOpacity(1015);
```

("Inactive" properties are introduced in version 1.6, SelOpacity is introduced in version 20)

If MainRVStyle⁶⁴⁷ is assigned, these properties provide access to the corresponding properties of MainRVStyle⁶⁴⁷.

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*¹⁷⁰ and *hotspots*¹⁷²,
- drawing a frame around the selected *breaks*¹⁶⁷,
- drawing background for the selected pictures¹⁶³ and controls¹⁶⁸,
- drawing background for selected table¹⁷³ cells.

In FireMonkey, **SelColor** and **SelTextColor** are used for drawing resizing handles and a resizing rectangle (for pictures¹⁶³ and controls¹⁶⁸), **SelColor** is used for drawing a table¹⁷³ sizing line.

Set **InactiveSelColor** and **InactiveSelTextColor** to *rvclNone*¹⁰³⁸ to hide a selection when the input focus shifts to another control.

TRichView must be linked with this TRVStyle (see TRichView.Style²⁵³).

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 *rvDefaultSelectionFill* is included in the Options²⁴⁰ 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*¹⁰³⁸
- SelTextColor: *rvclHighlightText*¹⁰³⁸
- InactiveSelColor: *rvclHighlight*¹⁰³⁸
- InactiveTextColor: *rvclHighlightText*¹⁰³⁸
- SelOpacity: 100000 (opaque)

See also properties:

- SelectionStyle⁶⁴⁹.

6.2.1.1.18 TRVStyle.InvalidPicture

Picture that will be inserted when reading RTF or RVF containing invalid or missing pictures.

property InvalidPicture: TRVPicture⁽⁹⁷⁵⁾;

(Introduced in version 1.7)

Default value:



(TBitmap)

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

See also properties of TCustomRichView:

- RVFWarnings⁽²⁵¹⁾.

6.2.1.1.19 TRVStyle.JumpCursor

Mouse cursor for hypertext indication.

property JumpCursor : TCursor

This cursor appears when user moves the mouse pointer to *hotspot*⁽¹⁷²⁾ or *hot-picture*⁽¹⁶⁶⁾ 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⁽⁷⁰⁴⁾).

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

crJump

See also:

- Cursors used by the components of TRichView family⁽¹⁴²⁾.

6.2.1.1.20 TRVStyle.LineSelectCursor

Mouse cursor for line selection

property LineSelectCursor : TCursor

(introduced in v1.8)

This cursor is displayed when the mouse pointer is above the left margin.

This cursor is used in the special text selection mode: line selection. Line selection is started if the user started selecting with the mouse from the left margin.

Line selection does not work for right-to-left documents⁽²²⁴⁾.

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

You can disable line selection if you set this property to *crNone* (-1).

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:*crRVFlipArrow***See also properties of TCustomRichView:**

- *LeftMargin* ⁽²³⁸⁾.

See also:

- Cursors used by the components of TRichView family ⁽¹⁴²⁾.

6.2.1.1.21 TRVStyle.LineWrapMode

Specifies how text lines are wrapped.

type

```
TRVLineWrapMode = (rvWrapAnywhere, rvWrapSimple, rvWrapNormal);
```

property *LineWrapMode*: TRVLineWrapMode

(introduced in version 13)

Value	Line wrapping algorithm
<i>rvWrapAnywhere</i>	Line break can occur in any place of text
<i>rvWrapSimple</i>	Line breaks inside text items ⁽¹⁶¹⁾ are added using line breaking rules. Line breaks between text items ⁽¹⁶¹⁾ can be added between any text items.
<i>rvWrapNormal</i>	Line breaks both inside and between text items are added using line breaking rules.

Line breaking rules are different for ANSI and Unicode ⁽¹³⁰⁾ 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

rvWrapAnywhere

The new version of RichViewAction:

- supports objects alignment ([screenshot](#)), headings ([screenshot](#)), styles of horizontal lines ([screenshot](#));
- includes new actions for pasting plain text, and for removing hyperlinks from the selected text.

[Read more about RichViewActions.](#)

Full source code is included in the [TRichView installation](#).

rvWrapSimple

The new version of RichViewAction:

- supports objects alignment ([screenshot](#)), headings ([screenshot](#)), styles of horizontal lines ([screenshot](#));
- includes new actions for pasting plain text, and for removing hyperlinks from the selected text.

[Read more about RichViewActions.](#)

Full source code is included in the [TRichView installation](#).

rvWrapNormal

The new version of RichViewAction:

- supports objects alignment ([screenshot](#)), headings ([screenshot](#)), styles of horizontal lines ([screenshot](#));
- includes new actions for pasting plain text, and for removing hyperlinks from the selected text.

[Read more about RichViewActions.](#)

Full source code is included in the [TRichView installation](#).

If `MainRVStyle`⁶⁴⁷ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁶⁴⁷.

Default value:

rvWrapNormal

6.2.1.1.22 TRVStyle.ListStyles

This is one of the key properties of TRVStyle: collection of paragraph list styles (bullets and numbering).

```
property ListStyles : TRVListInfos685 ;
```

Bullets/numbering (lists) are implemented in the following way:

1. TRVStyle has a collection of list styles (this property). Each list style⁷³¹ contains a collection of list levels (Levels⁷³² property). A list level⁷⁵⁰ defines the most of list properties.
2. List markers¹⁷⁴ 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:

- TextStyles⁶⁵⁴ ;
- ParaStyles⁶⁴⁸ .

See also types:

- TRVListInfos⁶⁸⁵ (collection of list styles);
- TRVListInfo⁷³¹ (item in the collection of list styles).

See also methods of TCustomRichView:

- SetListMarkerInfo³⁶² ,
- GetListMarkerInfo³⁰⁷ .

See also methods of TCustomRichViewEdit:

- ApplyListStyle⁴⁹⁰ ,
- RemoveLists⁵⁴² ,
- ChangeListLevels⁵⁰⁰ .

See also:

- List markers item type¹⁷⁴ .

6.2.1.1.23 TRVStyle.LiveSpellingColor

Color of live spelling's wavy underlines.

```
property LiveSpellingColor : TRVColor996 ;
```

(introduced in v1.9)

This color is used for underlines that mark misspelled words.

If MainRVStyle⁶⁴⁷ is assigned, this property provides access to the corresponding property of MainRVStyle⁶⁴⁷ .

Example:

Misspeled text

Default value:

`rvclRed`¹⁰³⁸

See also:

- Live spelling check in `TRichView`¹³².

6.2.1.1.24 TRVStyle.MainRVStyle

Defines the master `TRVStyle` component for this component.

property `MainRVStyle: TRVStyle`⁶³⁰;

(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`⁶⁵⁴
- `ParaStyles`⁶⁴⁸
- `ListStyles`⁶⁴⁶
- all events

This property allows:

- several editors can work with the same collection of style templates⁶⁵²
- 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 header³⁵⁶/footer³⁵⁶ (and style templates are used²⁵⁶) it must have `Style`²⁵³.`MainRVStyle` assigned equal to `Style`²⁵³ of the main editor.

 **ScaleRichView note:** `TSRichViewEdit` provides that `srv.RichViewEdit.Style = srv.RVHeader.Style.MainRVStyle = srv.RVFooter.Style.MainRVStyle`.

6.2.1.1.25 TRVStyle.PageBreakColor

Color of explicit (user-defined) page breaks in `TRichView` components.

property `PageBreakColor: TRVColor`⁹⁹⁶;

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`⁶⁶⁸ event.

Explicit page breaks are added by assigning `True` to `TRichView.PageBreaksBeforeItems`²⁴⁴[], or using `TRichViewEdit.InsertPageBreak`⁵²⁵.

`TRichView` must be linked with this `TRVStyle` (see `TRichView.Style`²⁵³).

If `MainRVStyle`⁶⁴⁷ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁶⁴⁷.

Default value:

`rvclBtnShadow`¹⁰³⁸

See also properties:

- `SoftPageBreakColor`⁶⁵¹.

See also events:

- `OnDrawPageBreak`⁶⁶⁸.

See also properties of TCustomRichView:

- Options⁽²⁴⁰⁾ (*rvoShowPageBreaks*).

6.2.1.1.26 TRVStyle.ParaStyles

This is one of the key properties of TRVStyle: collection of paragraph styles.

```
property ParaStyles : TParaInfos(682);
```

Paragraph styles define paragraph attributes in documents.

See also properties:

- TextStyles⁽⁶⁵⁴⁾;
- ListStyles⁽⁶⁴⁶⁾.

See also types:

- TParaInfos⁽⁶⁸²⁾ (collection of paragraph styles);
- TParaInfo⁽⁷¹⁸⁾ (item in collection of paragraph styles).

See also:

- Paragraphs in RichView⁽⁹⁵⁾.

6.2.1.1.27 TRVStyle.SelectionHandleKind

Defines visual appearance of selection handles for touch screens

```
property SelectionHandleKind: TRVSelectionHandleKind;
```

type

```
TRVSelectionHandleKind = (rvshkWedge, rvshkCircle);
```

(introduced in v15)

This property exists when the component is compiled in Delphi 2010 or newer.

Value	Appearance
<i>rvshkWedge</i>	
<i>rvshkCircle</i>	

rvshkCircle is not recommended, because it may overlap selected text in table cells;  you can use it in ScaleRichView, because TScaleRichViewEdit does not shift handles to fit in cells.

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

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 ⁽²⁵³⁾

6.2.1.1.28 TRVStyle.SelectionMode

Defines how the selection is guided by TRichView components.

property SelectionMode: TRVSelectionMode;

type

```
TRVSelectionMode =  
(rvsmChar, rvsmWord, rvsmParagraph);
```

(introduced in v 1.8)

Selection Mode	Meaning
<i>rvsmChar</i>	No special processing for selection: selection starts where the user pressed the left mouse button, selection ends where the user released it.
<i>rvsmWord</i>	Selecting by words. Selection is smart: you can still select a part of word by moving mouse pointer in backward direction. This mode affects only selecting by moving mouse.
<i>rvsmParagraph</i>	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 ⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle ⁽⁶⁴⁷⁾.

Default value:

rvsmWord

6.2.1.1.29 TRVStyle.SelectionStyle

Defines visual appearance of selection in TRichView components.

property SelectionStyle: TRVSelectionStyle;

type

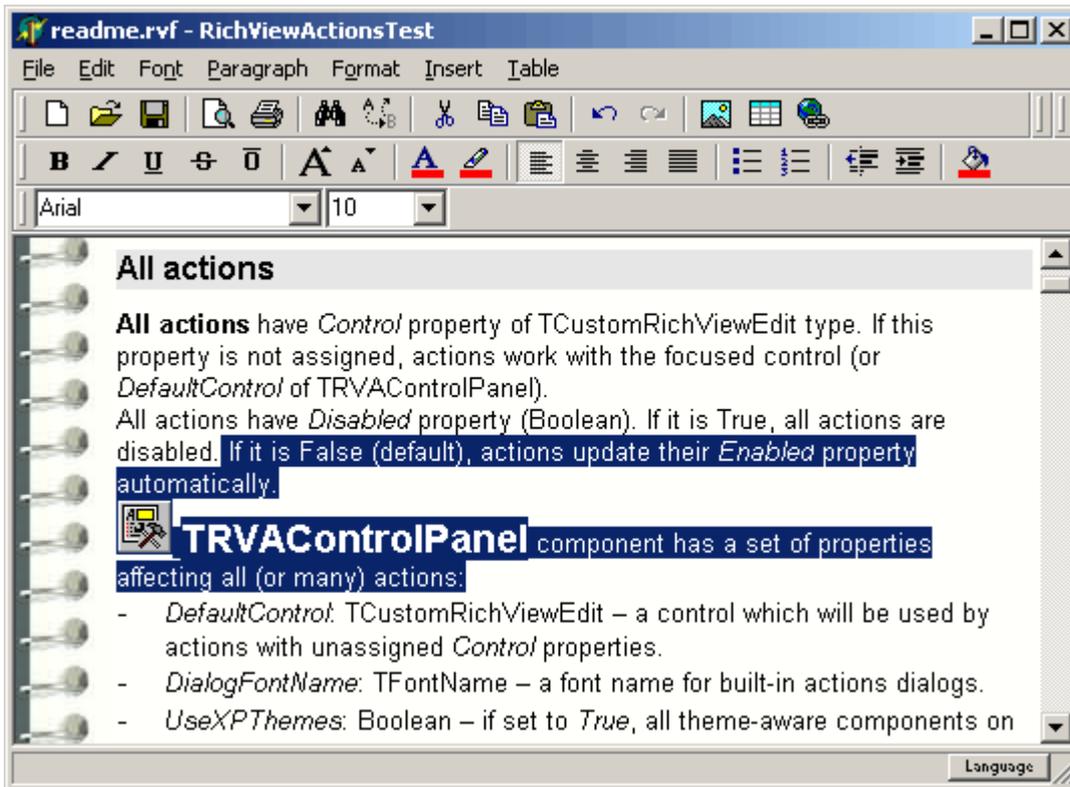
```
TRVSelectionStyle = (rvssItems, rvssLines);
```

(introduced in v 1.8)

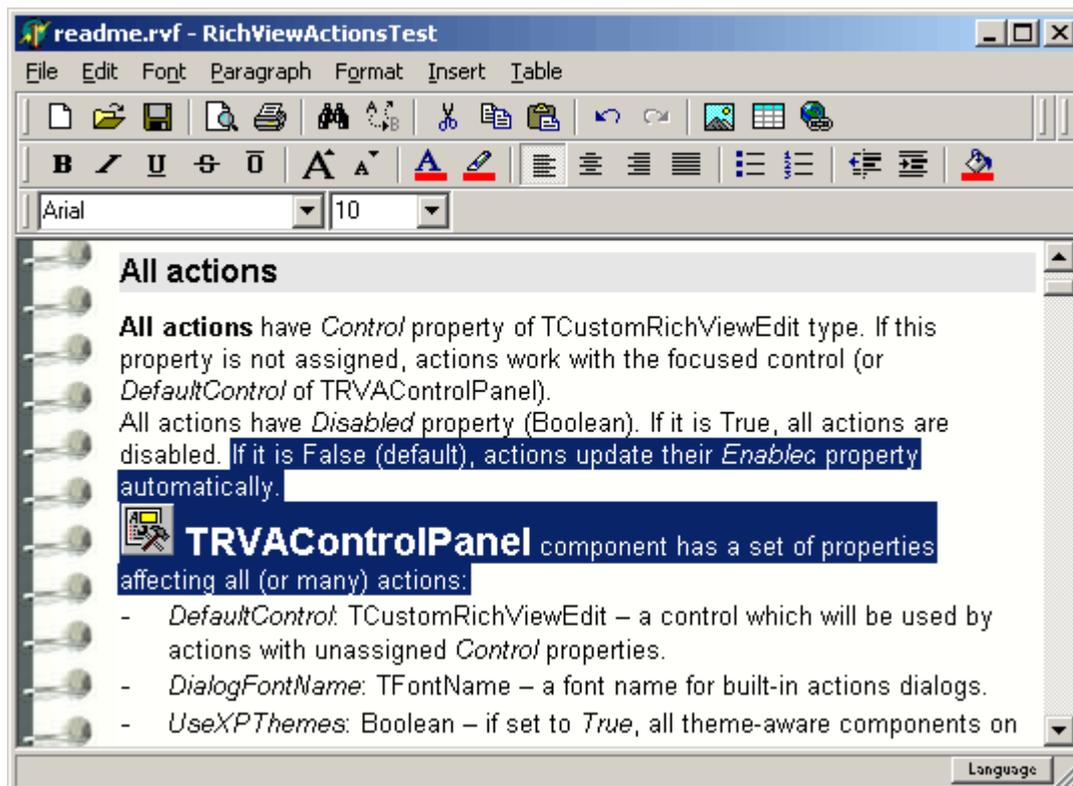
Selection Style	Meaning
<i>rvssItems</i>	Each item draws its selection separately.

Selection Style	Meaning
<i>rvssLines</i>	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 ⁽¹³²⁾ = <i>rvbdUnspecified</i> .

rvssItems:



rvssLines:



If `MainRVStyle`⁶⁴⁷ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁶⁴⁷.

Default value:

`rvsItems`

See also properties:

- `SelColor`, `SelTextColor`, `InactiveSelColor`, `InactiveSelTextColor`⁶⁴¹;
- `SelectionMode`⁶⁴⁹.

6.2.1.1.30 TRVStyle.SoftPageBreakColor

Color of soft page breaks in TRichView components.

property `PageBreakColor`: `TRVColor`⁹⁹⁶;

This color is used for drawing horizontal lines (with "folded paper" icon) specifying soft (automatic) page breaks. You can change this drawing in `OnDrawPageBreak`⁶⁸⁸ event.

TRichView must be linked with this TRVStyle (see `TRichView.Style`²⁵³). Soft page breaks displaying is turned on by `TRichView.AssignPageBreaks`²⁷⁷.

If `MainRVStyle`⁶⁴⁷ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁶⁴⁷.

Default value:

`rvclBtnFace`¹⁰³⁸

See also properties:

- `PageBreakColor`⁶⁴⁷.

See also events:

- OnDrawPageBreak⁽⁶⁶⁸⁾.

See also properties of TCustomRichView:

- Options⁽²⁴⁰⁾ (*rvoShowPageBreaks*).

6.2.1.1.31 TRVStyle.SpacesInTab

If set to positive value, TRichView components will replace TAB characters with the specified number of space characters.

property SpacesInTab: SmallInt;

This property is used in all methods of TRichView⁽²¹⁰⁾ and TRichViewEdit⁽⁴⁶¹⁾ for adding/inserting multiline text and RTF, or on typing.

TRichView must be linked with this TRVStyle (see TRichView.Style⁽²⁵³⁾).

If this property is 0, TABs are inserted as a special tabulator item type⁽¹⁶²⁾.

Note: default value is changed to 0 since v1.9

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

0

See also properties:

- DefTabWidth⁽⁶³⁶⁾.

6.2.1.1.32 TRVStyle.SpecialCharactersColor

Color of special characters.

property SpecialCharactersColor: TRVColor⁽⁹⁹⁶⁾;

(introduced in version 12)

if *rvoShowSpecialCharacters* is included in TRichView.Options⁽²⁴⁰⁾, this color is used for drawing marks denoting space characters and paragraph breaks.

If it equals to *rvclNone*⁽¹⁰³⁸⁾, a normal text color is used instead.

If MainRVStyle⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

rvclNone⁽¹⁰³⁸⁾

See also:

- RVVisibleSpecialCharacters⁽¹⁰⁶¹⁾ global typed constant
- FloatingLineColor⁽⁶³⁸⁾

6.2.1.1.33 TRVStyle.StyleTemplates

A collection of real styles, controlling text attributes from TextStyles⁽⁶⁵⁴⁾ and paragraph attributes from ParaStyles⁽⁶⁴⁸⁾.

property StyleTemplates : TRVStyleTemplateCollection⁽⁶⁸⁸⁾;

Using style templates is optional, they are used only for richview controls having `UseStyleTemplates`⁽²⁵⁶⁾ = `True`.

If `MainRVStyle`⁽⁶⁴⁷⁾ is assigned, this property returns `MainRVStyle`⁽⁶⁴⁷⁾.`StyleTemplates`.

See also

- `Styles and style templates`⁽⁹⁸⁾.

6.2.1.1.34 TRVStyle.TextBackgroundKind

Defines visual appearance of text background in TRichView components.

type

```
TRVTextBackgroundKind = (rvtbkSimple, rvtbkItems, rvtbkLines);
```

property `TextBackgroundKind`: `TRVTextBackgroundKind`;

(introduced in v14)

Selection Style	Background is drawn...	Background height =
<i>rvtbkItems</i>	before text	item height
<i>rvtbkSimple</i>	together with text	item height
<i>rvtbkLines</i>	before text	line height

Example:

Painful zombies quickly watch a jinxed graveyard
rvtbkItems

Painful zombies quickly watch a jinxed graveyard
rvtbkSimple

Painful zombies quickly watch a jinxed graveyard
rvtbkLines

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`⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁽⁶⁴⁷⁾.

Default value:

rvtbkItems

See also:

- `OnDrawTextBack`⁽⁶⁷³⁾

See also properties of text style:

- `BackColor`⁽⁶⁹⁸⁾
- `HoverBackColor`⁽⁷⁰²⁾;

6.2.1.1.35 TRVStyle.TextEngine

Specifies which set of functions are used for text measuring and drawing.

type

```
TRVTextEngine = (rvteWindows, rvteUniscribe);
```

property TextEngine: TRVTextEngine;

(introduced in v16)

Value	Meaning	Comments
<i>rvteWindows</i>	Windows API	<p>Pros:</p> <ul style="list-style-type: none"> • faster, especially if the support of bidirectional text is turned off. <p>Cons:</p> <ul style="list-style-type: none"> • 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).
<i>rvteUniscribe</i>	Uniscribe	<p>Pros:</p> <ul style="list-style-type: none"> • reliable processing of complex scripts and "exotic" characters; • acceptable results for right-to-left text even if support of bidirectional text is turned off <p>Cons:</p> <ul style="list-style-type: none"> • usually slower

Reformat documents in linked TRichView controls after changing value of this property.

If `MainRVStyle`⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of `MainRVStyle`⁽⁶⁴⁷⁾.

Note: this property is not available in FireMonkey version.

Default value:

rvteUniscribe

6.2.1.1.36 TRVStyle.TextStyles

This is one of key properties of TRVStyle: a collection of text attributes (styles).

property TextStyles : TFontInfos⁽⁶⁷⁵⁾;

Text styles define fonts and other text attributes in documents.

The first text style in the collection (`TextStyles[0]`) has some additional meaning:

- `breaks`⁽⁸⁵⁾ with `color = c/None` (default value) have the color of this style;
- `Charset`⁽⁷⁰⁰⁾ 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⁽⁷⁰⁵⁾ for this style;
- this style can be hypertext, but it is not recommended.

See also properties:

- ParaStyles⁽⁶⁴⁸⁾;
- ListStyles⁽⁶⁴⁶⁾.

See also types:

- TFontInfos⁽⁶⁷⁵⁾ (collection of text styles);
- TFontInfo⁽⁷¹²⁾ (item in collection of text styles).

6.2.1.1.37 TRVStyle.Units

Defines units of measurement for properties of TRVStyle and linked⁽²⁵³⁾ TRichView controls.

property Units: TRVStyleUnits⁽¹⁰²⁷⁾

(introduced in version 13)

All values of type TRVStyleLength⁽¹⁰²⁷⁾ are measured in units specified in this property.

If Units = `rvstuPixels`, DPI (dots per inch) of values is defined in UnitsPixelsPerInch⁽⁶⁵⁵⁾ 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*⁽²⁸⁰⁾ methods of all TRichView⁽²¹⁰⁾ components linked⁽²⁵³⁾ to this TRVStyle, then call one of ConvertTo*⁽⁶⁵⁷⁾ 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⁽⁶⁴⁷⁾ is assigned, this property provides access to the corresponding property of MainRVStyle⁽⁶⁴⁷⁾.

Default value:

`rvstuPixels`

See also properties:

- Units of measurement in TRichView⁽¹³⁹⁾ (contains a list of properties measured in Units)

6.2.1.1.38 TRVStyle.UnitsPixelsPerInch

Specifies number of pixels in one inch.

property UnitsPixelsPerInch: Integer;

(introduced in version 18)

This property is applied to:

- values (of TRVStyleLength⁽¹⁰²⁷⁾ type) measured in Units⁽⁶⁵⁵⁾: TRVStyleUnits⁽¹⁰²⁷⁾, if Units = `rvstuPixels`;
- values (of TRVLength⁽¹⁰¹⁵⁾ type) measured in TRVUnits⁽¹⁰³²⁾, if Units = `rvuPixels`;
- values measured in pixels (of TRVPixelLength⁽¹⁰¹⁹⁾ type);
- sizes of pictures⁽¹⁶³⁾ 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 on the screen;
- this setting does not affect text size.

The recommended way of zooming is changing `TRichView(210).DocumentPixelsPerInch(233)` property.

If `MainRVStyle(647)` is assigned, this property provides access to the corresponding property of `MainRVStyle(647)`.

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(139)`

6.2.1.1.39 TRVStyle.UseSound

Defines whether `RichViewEdit(461)` 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⁽⁴⁸⁰⁾ or a protected⁽⁷⁰⁵⁾ text.

If `MainRVStyle(647)` is assigned, this property provides access to the corresponding property of `MainRVStyle(647)`.

Default value:

`True`

6.2.1.2 Methods

In TRVStyle

`AddTextStyle(657)` (deprecated)
`Create(657)`

ConvertToPixels, ConvertToTwips, ConvertToEMU, ConvertToDifferentUnits⁶⁵⁷
 DeleteTextStyle⁶⁵⁸ (deprecated)
 Destroy⁶⁵⁸
 GetAsPixels, GetAsStandardPixels, GetAsHTMLPixels, GetAsTwips, GetAsDifferentUnits,
 GetAsRVUnits⁶⁵⁸
 PixelsToUnits, StandardPixelsToUnits, HTMLPixelsToUnits, TwipsToUnits,
 DifferentUnitsToUnits, RVUnitsToUnits⁶⁵⁹
 FindParaStyle, FindTextStyle⁶⁶¹
 GetListStyleClass⁶⁶⁰
 GetParaStyleClass⁶⁶¹
 GetTextStyleClass⁶⁶¹
 LoadINI⁶⁶¹
 LoadReg⁶⁶²
 SaveCSS⁶⁶²
 SaveCSSToStream⁶⁶³
 SaveINI⁶⁶³
 SaveReg⁶⁶⁴

6.2.1.2.1 TRVStyle.AddTextStyle

```
function AddTextStyle: Integer;
```

Obsolete. In order to add item in collection of text styles, use RVStyle.TextStyles.Add⁶⁷⁸

6.2.1.2.2 TRVStyle.Create

A constructor, creates a new TRVStyle component.

```
constructor Create(AOwner: TComponent); override;
```

Use Create to instantiate TRVStyle component at runtime. TRVStyle components placed on forms at design time are created automatically. Specify the owner of the new RVStyle using the AOwner parameter.

6.2.1.2.3 TRVStyle.ConvertTo*

The methods convert all properties of this TRVStyle component to new units of measurement.

```
procedure ConvertToPixels;
```

```
procedure ConvertToTwips;
```

```
procedure ConvertToEMU;
```

```
procedure ConvertToDifferentUnits(NewUnits: TRVStyleUnits1027);
```

(introduced in versions 13, 18)

The methods convert all properties of this TRVStyle (including all sub-properties) from Units⁶⁵⁵ to:

- pixels (1 pixel = 1/UnitsPixelsPerInch⁶⁵⁵ of an inch)
- twips,
- EMU
- NewUnits (pixels, twips, or EMU),

respectively.

After the conversion, a new value is assigned to Units⁶⁵⁵ property.

If there is a document in TRichView control linked to this TRVStyle component, this document must be converted⁽²⁸⁰⁾ before calling this method.

See also:

- Units of measurement in TRichView⁽¹³⁹⁾

6.2.1.2.4 TRVStyle.DeleteTextStyle

```
procedure DeleteTextStyle(Index: Integer);
```

Obsolete. In order to delete the i-th item from text styles, use TRVStyle.TextStyles⁽⁶⁵⁴⁾[i].Free

6.2.1.2.5 TRVStyle.Destroy

A destructor, frees an instance of a TRVStyle component,

```
destructor Destroy; override;
```

Destroy seldom needs to be called. When TRVStyle is constructed without an owner by the Create method, Free should be called to release memory and dispose the object. Free checks to see if the pointer is nil before calling Destroy.

6.2.1.2.6 TRVStyle.GetAs*

The methods return **Value** converted from Units⁽⁶⁵⁵⁾ to different units of measurement.

```
function GetAsPixels(Value: TRVStyleLength(1027);
  PixelsPerInch, MinNonZeroValue: TRVCoord(998)): TRVCoord(998);
function GetAsHTMLPixels(Value: TRVStyleLength(1027)): TRVPixel96Length(1018);
function GetAsStandardPixels(Value: TRVStyleLength(1027)): TRVPixelLength(1019);

function GetAsTwips(Value: TRVStyleLength(1027)): Integer;
function GetAsEMU(Value: TRVStyleLength(1027)): Integer;
function GetAsDifferentUnits(Value: TRVStyleLength(1027);
  Units: TRVStyleUnits(1027)): TRVStyleLength(1027);
function GetAsRVUnits(Value: TRVStyleLength(1027);
  RVUnits: TRVUnits(1032)): TRVLength(1015);
```

(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 ⁽⁶⁵⁵⁾ of an inch.
GetAsTwips	Twips.

Method	Converts to
	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 ⁽⁶⁵⁵⁾ is used as DPI.
GetAsRVUnits	RVUnits (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch ⁽⁶⁵⁵⁾ 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⁽⁶⁵⁹⁾ methods;
- ConvertTo*⁽⁶⁵⁷⁾ methods;
- Units of measurement in TRichView⁽¹³⁹⁾

6.2.1.2.7 TRVStyle.*ToUnits

The methods return **Value** converted from different units of measurement to Units⁽⁶⁵⁵⁾.

```

function PixelsToUnits(Value: TRVCoord(998);
  PixelsPerInch: Integer): TRVStyleLength(1027);
function HTMLPixelsToUnits(Value: TRVPixel96Length(1018)): TRVStyleLength(1027);
function StandardPixelsToUnits(
  Value: TRVPixelLength(1019)): TRVStyleLength(1027);

function TwipsToUnits(Value: Integer): TRVStyleLength(1027);
function EMUToUnits(Value: Integer): TRVStyleLength(1027);
function DifferentUnitsToUnits(Value: TRVStyleLength(1027);
  Units: TRVStyleUnits(1027)): TRVStyleLength(1027);
function RVUnitsToUnits(Value: TRVLength(1015);
  RVUnits: TRVUnits(1032)): TRVStyleLength(1027);

```

(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.

Method	Converts from
	1 pixel = 1/96 of an inch.
StandardPixelsToUnits	Pixels. 1 pixel = 1/UnitsPixelsPerInch ⁽⁶⁵⁵⁾ 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 ⁽⁶⁵⁵⁾ is used as DPI.
RVUnitsToUnits	RVUnits (specified in the parameter). When converting to/from pixels, UnitsPixelsPerInch ⁽⁶⁵⁵⁾ 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*⁽⁶⁵⁸⁾ methods;
- ConvertTo*⁽⁶⁵⁷⁾ methods;
- Units of measurement in TRichView⁽¹³⁹⁾.

6.2.1.2.8 TRVStyle.GetListStyleClass

Returns the class type representing a style of paragraph lists (bullets and numbering).

type

```
TRVListInfoClass = class of TRVListInfo(731);
```

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⁽⁷³¹⁾) and use it in this descendant component.

See also:

- GetParaStyleClass⁽⁶⁶¹⁾;
- GetTextStyleClass⁽⁶⁶¹⁾.

6.2.1.2.9 TRVStyle.GetParaStyleClass

Returns the class type representing a style of paragraphs.

type

```
TRVParaInfoClass = class of TParaInfo(718);
```

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⁽⁷¹⁸⁾) and use it in this descendant component.

See also:

- GetListStyleClass⁽⁶⁶⁰⁾;
- GetTextStyleClass⁽⁶⁶¹⁾.

6.2.1.2.10 TRVStyle.GetTextStyleClass

Returns the class type representing a style of text.

type

```
TRVFontInfoClass = class of TFontInfo(712);
```

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⁽⁷¹²⁾) and use it in this descendant component.

See also:

- GetListStyleClass⁽⁶⁶⁰⁾;
- GetParaStyleClass⁽⁶⁶¹⁾.

6.2.1.2.11 TRVStyle.LoadINI

Loads all properties of TRVStyle component from ini-file

```
procedure LoadINI(const FileName, Section: String);
```

Parameters:

FileName – name of the ini-file;

Section – section of the ini-file.

See also:

- SaveINI⁽⁶⁶³⁾;
- LoadReg⁽⁶⁶²⁾.

6.2.1.2.12 TRVStyle.FindTextStyle, FindParaStyle

The methods search for a style like the style specified in the parameter, and add it if necessary.

```
function FindTextStyle(TextStyle: TFontInfo(712)): Integer;
```

```
function FindParaStyle(ParaStyle: TParaInfo(727)): Integer;
```

(introduced in version 14)

The methods are equivalent to:

```
TextStyles(654).FindSuchStyleEx(681)(0, TextStyle, RAllFontInfoProperties(1008))
ParaStyles(648).FindSuchStyleEx(685)(0, ParaStyle, RAllParaInfoProperties(1017))
```

6.2.1.2.13 TRVStyle.LoadReg

Loads all properties of TRVStyle component from the Registry (for Delphi4+).

```
procedure LoadReg(const BaseKey: String);
```

(Introduced in version 1.2)

Parameter:

BaseKey – the name of the HKEY_CURRENT_USER subkey to loads data from. LoadReg appends 'RVStyle' to this subkey. For example, if **BaseKey** is 'Software\My Company\My Program', it loads data from the 'Software\My Company\My Program\RVStyle' key.

Warning: these data in this section (BaseKey+'RVStyle') must exists, otherwise all properties will be reset to default values.

See also:

- SaveReg⁽⁶⁶⁴⁾;
- LoadINI⁽⁶⁶¹⁾.

6.2.1.2.14 TRVStyle.SaveCSS

Saves text and paragraph styles as CSS (Cascading Style Sheets) file.

type

```
TRVSaveCSSTOption = (
  rvcssOnlyDifference, rvcssIgnoreLeftAlignment,
  rvcssNoDefCSSStyle, rvcssDefault0Style,
  rvcssMarkersAsText);
TRVSaveCSSTOptions = set of TRVSaveCSSTOption;
```

```
function SaveCSS(const FileName: TRVUnicodeString(1032);
  AOptions: TRVSaveCSSTOptions;
  Encoding: TRVHTMLEncoding(1011)): 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⁽²¹⁰⁾.HTMLSaveProperties⁽²³⁷⁾.ExternalCSSFileName⁽⁴³²⁾.

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
<i>rvcssNoDefCSSStyle</i>	Corresponds to <i>rvcssNoDefCSSStyle</i> in TRichView.HTMLProperties.CSSOptions ⁴³⁰ . It's not recommended to use this option.
<i>rvcssDefault0Style</i>	Properties of TextStyles ⁶⁵⁴ [0] are not saved. Corresponds to TRichView.HTMLProperties.Default0Style ⁴³¹ .
<i>rvcssMarkersAsText</i>	Lists will be saved as text. TRichView.HTMLProperties.ListMarkerSavingType ⁴³⁶ = <i>rvhtmlmstAsText</i> .

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 files¹²⁷.

6.2.1.2.15 TRVStyle.SaveCSSToStream

The same as SaveCSS⁶⁶², but accepts **Stream** parameter

```
procedure SaveCSSToStream(Stream: TStream; Options: TRVSaveCSOptions);  
(introduced in v1.1.4)
```

6.2.1.2.16 TRVStyle.SaveINI

Saves all properties of TRVStyle component in the ini-file

```
procedure SaveINI(const FileName, Section: String);
```

Parameters:

FileName – name of the ini-file;

Section – section of the ini-file

Warning: the section is cleared before saving.

Alternative way to store/load TRVStyle: using TStream.WriteComponent and ReadComponent.

See also:

- LoadINI⁶⁶¹;
- SaveReg⁶⁶⁴;
- SaveCSS⁶⁶².

6.2.1.2.17 TRVStyle.SaveReg

Saves all properties of component to the Registry (for Delphi4+)

```
procedure SaveReg(const BaseKey: String);
```

(Introduced in version 1.2)

Parameter:

BaseKey – the name of the HKEY_CURRENT_USER subkey to saves data to. SaveReg appends 'RVStyle' to this subkey. For example, if **BaseKey** is 'Software\My Company\My Program', the method saves data to the 'Software\My Company\My Program\RVStyle' key.

Warning: all data in this key (BaseKey+'RVStyle') will be removed before saving!

The saving format is the same as in SaveINI⁶⁶³.

See also:

- LoadReg⁶⁶²;
- SaveINI⁶⁶³.

6.2.1.3 Events

In TRVStyle

- OnAfterApplyStyle⁶⁶⁴
- OnApplyStyle⁶⁶⁵
- OnApplyStyleColor⁶⁶⁶
- OnDrawCheckpoint⁶⁶⁷
- OnDrawPageBreak⁶⁶⁸
- OnDrawParaBack⁶⁷⁰
- OnDrawStyleText⁶⁷¹
- OnDrawTextBack⁶⁷³
- OnStyleHoverSensitive⁶⁷⁴

6.2.1.3.1 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: TRVFMXCanvas961;  
    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`⁽⁶⁶⁵⁾, but occurs after the default procedure. This event does not occur if `OnApplyStyle`⁽⁶⁶⁵⁾ returned *False* in `DoDefault` parameter.

Input parameters:

Sender – TRVStyle generating this event;

Canvas – canvas for applying font attributes to.

See also:

- `OnApplyStyleColor`⁽⁶⁶⁶⁾;
- `OnDrawStyleText`⁽⁶⁷¹⁾.

See also properties:

- `TextStyles`⁽⁶⁵⁴⁾.

6.2.1.3.2 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(1024);
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3; modified in v18)

FireMonkey:

type

```
TRVApplyStyleEvent =
  procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas(961);
    StyleNo: Integer; PSaD: PRVScreenAndDevice(1024);
    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`⁽¹⁰²⁷⁾ 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**⁶⁶⁴ event.

See also:

- **OnApplyStyleColor**⁶⁶⁶;
- **OnDrawStyleText**⁶⁷¹.

See also properties:

- **TextStyles**⁶⁵⁴.

6.2.1.3.3 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: TRVTextDrawStates1030;
    Printing: Boolean; ColorMode: TRVColorMode997;
    DarkMode: Boolean;
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3, changed in version 23)

FireMonkey:

type

```
TRVApplyStyleColorEvent =
  procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas961;
    StyleNo: Integer; DrawState: TRVTextDrawStates1030;
    Printing: Boolean; ColorMode: TRVColorMode997;
    DarkMode: Boolean;
    var DoDefault: Boolean) of object;
```

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 TRVTextDrawStates¹⁰³⁰.

DoDefault is equal to *True*.

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⁽⁶⁶⁵⁾;
- OnDrawStyleText⁽⁶⁷¹⁾.

See also properties:

- TextStyles⁽⁶⁵⁴⁾.

6.2.1.3.4 TRVStyle.OnDrawCheckpoint

Occurs when drawing *checkpoint*⁽⁸⁷⁾.

property OnDrawCheckpoint: TRVDrawCheckpointEvent;

VCL and LCL:**type**

```
TRVDrawCheckpointEvent =
  procedure (Sender: TRVStyle; Canvas: TCanvas;
    X,Y: TRVCoord(998); ItemNo: Integer; XShift: TRVCoord(998);
    RaiseEvent: Boolean; Control: TControl;
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

FireMonkey:**type**

```
TRVDrawCheckpointEvent =
  procedure (Sender: TRVStyle; Canvas: TRVFMXCanvas(961);
    X,Y: TRVCoord(998); ItemNo: Integer; XShift: TRVCoord(998);
    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⁽²⁴⁰⁾.

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⁽⁹⁵⁷⁾ 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⁽⁶³⁴⁾, 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(993);
  CheckpointTag: TRVTag(1029);
  CheckpointName: String;
begin
  if ItemNo >= 0 then
  begin
    CheckpointData :=
      TCustomRVFormattedData(957)(Sender.RVData).GetItemCheckpoint(298)(
        ItemNo);
    TCustomRVFormattedData(Sender.RVData).GetCheckpointInfo(291)(
      CheckpointTag, CheckpointName, RaiseEvent);
    ...
  end;
end;
```

See also properties:

- CheckpointColor, CheckpointEvColor⁽⁶³⁴⁾.

See also:

- Checkpoints overview⁽⁸⁷⁾.

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.1.3.5 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: TRVCoord998;
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: TRVFMXCanvas961;
    Y, XShift: TRVCoord998;
    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).Options240`.

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 `TCustomRVFormattedData957` before using.

Sender.ItemNo: Integer, index of the item in **Sender.RVData**.

These parameters define one of the following:

- item starting a new page;
- table⁸⁴⁶ with page breaks between rows.

Output parameter:

DoDefault – set to *False* to prevent default drawing. The default procedure draws horizontal line with "dog-ear" effect. The line color is either `PageBreakColor647` or `SoftPageBreakColor651`, depending on **PageBreakType**.

Using **Sender.RVData** you can get additional information about the item:

For example:

```
uses ... , CRVFDData;
...
var ParaNo: Integer;
begin
  ParaNo := TCustomRVFormattedData957 (Sender.RVData).GetItemPara301 (
    Sender.ItemNo);
  ...
end;
```

See also properties:

- PageBreakColor⁶⁴⁷;
- SoftPageBreakColor⁶⁵¹.

See also properties of TCustomRichView²¹⁰:

- PageBreaksBeforeItem²⁴⁴.

See also methods of TCustomRichView²¹⁰:

- AssignSoftPageBreaks²⁷⁷.

See also methods of TCustomRichViewEdit⁴⁶¹:

- InsertPageBreak⁵²⁵;
- RemoveCurrentPageBreak⁵⁴².

See also properties of table⁸⁴⁶:

- PrintOptions⁸⁶³.

See also properties of table row⁹⁰⁹:

- PageBreakBefore⁹¹⁰.

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.1.3.6 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: TRVCoordRect998;
    var DoDefault: Boolean) of object;
```

FireMonkey:

type

```
TRVDrawParaRectEvent =
  procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas961;
    ParaNo: Integer; ARect: TRVCoordRect998;
    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 (ParaStyles⁶⁴⁸).

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.RVData: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData⁹⁵⁷ 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:

- ParaStyles⁶⁴⁸;
- TParaInfo⁷¹⁸.Background⁷²¹.

See also:

- RichView Paragraphs⁹⁵.

Demo project:

- Demos*\Assorted\Custom Draw\CustomDraw\

6.2.1.3.7 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: TRVStyle630;
  const s: TRVUnicodeString1032; ACanvas: TCanvas; StyleNo: Integer;
  SpaceAtLeft, ALeft, ATop, AWidth, AHeight
  {$IFDEF RVUSEBASELINE}, BaseLine{$ELSE}
  {$IFDEF RVHIGHFONTS}, DrawOffsY{$ENDIF}
  {$ENDIF}: TRVCoord998;
  TextWidth: TRVCoord998; SpaceCount: Integer;
  DrawState: TRVTextDrawStates1030;
  Printing, PreviewCorrection, ForMetafile: Boolean;
  ColorMode: TRVColorMode997; DarkMode: Boolean;
```

```

DefBiDiMode: TRVBidiMode993;
RefCanvas: TCanvas;
LeftNoExpIndex, RightNoExpIndex: Integer;
PSaD: PRVScreenAndDevice1024;
var DoDefault: Boolean) of object;

```

(introduced in version 1.3; new states in TRVTextDrawStates¹⁰³⁰ since 1.4; changed in v16, v18, v23)

FireMonkey:

type

```

TRVDrawStyleTextEvent = procedure (Sender: TRVStyle630;
  const s: TRVUnicodeString1032; ACanvas: TRVFMXCanvas961;
  StyleNo: Integer;
  SpaceAtLeft, ALeft, ATop, AWidth, AHeight
  {$IFDEF RVUSEBASELINE}, BaseLine{$ELSE}
  {$IFDEF RVHIGHFONTS}, DrawOffsY{$ENDIF}
  {$ENDIF}: TRVCoord998;
  TextWidth: TRVCoord998; SpaceCount: Integer;
  DrawState: TRVTextDrawStates1030;
  Printing, PreviewCorrection, ForMetafile: Boolean;
  ColorMode: TRVColorMode997; DarkMode: Boolean;
  DefBiDiMode: TRVBidiMode993;
  RefCanvas: TCanvas;
  LeftNoExpIndex, RightNoExpIndex: Integer;
  PSaD: PRVScreenAndDevice1024;
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 TFontInfo⁷¹².Draw⁷¹⁶. You can call **Sender.TextStyles**⁶⁵⁴[**StyleNo**].Draw(..) passing the even parameters.

"Hidden" input parameters:

Sender.RVData: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData⁹⁵⁷ 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: TRVTag1029;
...
ItemTag := TCustomRVFormattedData957(Sender.RVData).GetItemTag302(
  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:

- OnApplyStyle⁶⁶⁵;
- OnApplyStyleColor⁶⁶⁶;
- OnDrawTextBack⁶⁷³.

See also properties:

- TextStyles⁶⁵⁴.

Demo projects:

- Demos*\Assorted\Custom Draw\

6.2.1.3.8 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: TRVCoord998;
    DrawState: TRVTextDrawStates1030;
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

FireMonkey:

type

```
TRVDrawTextBackEvent =
  procedure (Sender: TRVStyle; ACanvas: TRVFMXCanvas961;
    StyleNo: Integer; ALeft, ATop, AWidth, AHeight: TRVCoord998;
    DrawState: TRVTextDrawStates1030;
    var DoDefault: Boolean) of object;
```

You can use this event to modify drawing of text background for some or all text styles.

Unlike OnDrawStyleText⁶⁷¹, this event is called only once for each drawing item (OnDrawStyleText⁶⁷¹ can be called several times if part of text item is selected). In paragraphs having Alignment⁷¹⁹=*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`⁽⁶⁵⁴⁾ collection.

Left, Top, Width, Height – rectangle where to paint.

DrawState may include `rvtsControlFocused` and `rvtsHover`, see , see `TRVTextDrawStates`⁽¹⁰³⁰⁾.

DoDefault is equal to `True`.

"Hidden" input parameters:

Sender.RVData: `TPersistent`, document containing this paragraph. It should be typecasted to `TCustomRVFormattedData`⁽⁹⁵⁷⁾ 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(1029);
...
ItemTag := TCustomRVFormattedData(957)(Sender.RVData).GetItemTag(302)(
    Sender.ItemNo);
```

Output parameters:

DoDefault – set to `False` to prevent default drawing. It is supported only if `TextBackgroundKind`⁽⁶⁵³⁾ `<>rvtbkSimple`.

See also events:

- `OnDrawStyleText`⁽⁶⁷¹⁾
- `OnApplyStyle`⁽⁶⁶⁵⁾;
- `OnApplyStyleColor`⁽⁶⁶⁶⁾;

See also properties:

- `TextStyles`⁽⁶⁵⁴⁾.

Demo projects:

- `Demos*\Assorted\Custom Draw\`

6.2.1.3.9 TRVStyle.OnStyleHoverSensitive

In this event you can define whether `TRichView` should redraw itself when the mouse pointer moves in or out of the area of text of the **StyleNo**-th style.

```
property OnStyleHoverSensitive: TRVStyleHoverSensitiveEvent;
```

type

```
TRVStyleHoverSensitiveEvent =
    procedure (Sender: TRVStyle;
        StyleNo: Integer;
        var Sensitive: Boolean) of object;
```

(introduced in version 1.3)

This event occurs only for hypertext⁽⁷¹⁴⁾ styles.

Input parameters:

Sender – TRVStyle generating the event.

StyleNo – index of text style in TextStyles⁽⁶⁵⁴⁾ collection.

Sensitive – *True* if for the given text style (HoverEffects⁽⁷⁰³⁾ =[]) or (HoverBackColor⁽⁷⁰²⁾ <>BackColor⁽⁶⁹⁸⁾) or (HoverColor⁽⁷⁰²⁾ <>c/None) or (HoverUnderlineColor⁽⁷⁰³⁾ <>c/None), or if HoverColor⁽⁶⁴¹⁾ <>c/None. *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⁽⁶⁷¹⁾;
- OnDrawTextBack⁽⁶⁷³⁾.

See also properties:

- TextStyles⁽⁶⁵⁴⁾.

6.2.1.4 Classes of properties - Style collections

Classes defining formatting

- TFontInfos⁽⁶⁷⁵⁾ – collection of items representing character formatting, a type of TRVStyle⁽⁶³⁰⁾.TextStyles⁽⁶⁵⁴⁾ property
- TParaInfos⁽⁶⁸²⁾ – collection of items representing paragraph formatting, a type of TRVStyle.ParaStyles⁽⁶⁴⁸⁾ property
- TRVListInfos⁽⁶⁸⁵⁾ – collection of items representing paragraph lists (bullets and numbering), a type of TRVStyle.ListStyles⁽⁶⁴⁶⁾ property

"Real styles"

- TRVStyleTemplateCollection⁽⁶⁸⁸⁾ – collection of items representing named styles, a type of TRVStyle.StyleTemplates⁽⁶⁵²⁾ property

6.2.1.4.1 TFontInfos

Collection of text attributes (styles). Class of TRVStyle.TextStyles⁽⁶⁵⁴⁾ property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
TFontInfos = class(TCustomRVInfos)
```

Hierarchy

TObject

TPersistent

TCollection

TCustomRVInfos

Main Properties

Items⁶⁷⁷ – array of TFontInfo⁷¹².

InvalidItem⁶⁷⁶ is returned when accessing item which is out of index range.

Methods for Text Style Search

- FindStyleWithFontStyle⁶⁸¹
- FindStyleWithFontSize⁶⁸⁰
- FindStyleWithColor⁶⁷⁹
- FindStyleWithFontName⁶⁸⁰
- FindSuchStyle, FindSuchStyleEx⁶⁸¹
- FindStyleWithFont⁶⁷⁹
- FindStyleWithCharset⁶⁷⁹

See Also

- TParaInfos⁶⁸² – class of collection of paragraph attributes (styles);
- TRVListInfos⁶⁸⁵ – class of collection of list attributes (i.e. styles of bullets and numbering).

6.2.1.4.1.1 Properties

In TFontInfos

InvalidItem⁶⁷⁶

Items⁶⁷⁷

Derived from TCollection

Count

This item is returned when accessing Items⁶⁷⁷[ItemNo], if (ItemNo<0) or (ItemNo>=TFontInfos.Count)

property InvalidItem: TFontInfo⁷¹²;

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value:

Default value has all properties Items⁶⁷⁷[0], but red background⁶⁹⁸ and white text color⁷⁰¹.

See also:

- TParaInfos.InvalidItem⁶⁸³.

Provides indexed access to the items in the collection of text attributes (styles)

```
property Items[Index: Integer]: TFontInfo712;
```

This is the default property of TFontInfos, so you can write

```
MyRVStyle.TextStyles654[i]
```

instead of

```
MyRVStyle.TextStyles.Items[i]
```

Many methods of components inherited from TCustomRichView²¹⁰ have StyleNo parameter, which is an index in TFontInfos.Items collection (more specifically, in TCustomRichView.Style²⁵³.TextStyles⁶⁵⁴.Items).

Example:

```
MyRichView.AddNL270('Hello', 0, 0);
```

Unlike normal collections, when accessing item with invalid Index (for example, Index<0 or Index>=Count), no exception occurs. InvalidItem⁶⁷⁶ is returned instead.

Indicates the number of pixels that make up a logical inch in the vertical direction.

```
PixelsPerInch: Integer;
```

DEPRECATED!

Use TRichView²¹⁰.DocumentPixelsPerInch²³³ instead of this property.

This property affects how text is displayed in components inherited from TCustomRichView²¹⁰ and TRVReportHelper⁸⁰⁴ (this property is ignored by TRVPrint⁷⁵⁹).

Note: this property is not available in FireMonkey version

Default value:

0

6.2.1.4.1.2 Methods

In TFontInfos

```
Add678  
Create678  
FindStyleWithCharset679  
FindStyleWithColor679  
FindStyleWithFont679  
FindStyleWithFontName680  
FindStyleWithFontSize680  
FindStyleWithFontStyle681  
FindSuchStyle681  
FindSuchStyleEx681
```

Derived from TCustomRVInfos

```
AssignTo678
```

You can also use methods derived from TCollection

Adds a new style in the collection of text styles

```
function Add: TFontInfo(712);
```

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⁽⁶⁷⁷⁾ array.

Return value:

Created collection item.

See also:

- TFontInfo.Create⁽⁷¹⁶⁾.

Adds a new item in the collection of text styles, initializes it.

```
function AddFont(Name: TFontName; Size: Integer;
  Color, BackColor: TRVColor(936); Style: TFontStyles): TFontInfo(712);
```

Please use Add⁽⁶⁷⁸⁾ 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(936); Style: TFontStyles;
  Charset: TRVFontCharset(1008)): TFontInfo(712);
```

Please use Add⁽⁶⁷⁸⁾ 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⁽⁶⁹⁶⁾) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.TextStyles(654));
```

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⁽⁶⁵⁴⁾ property.

Searches for style with the specified **Charset**.

```
function FindStyleWithCharset(BaseStyle: Integer;  
    Charset: TRVFontCharset1008): Integer;
```

(Introduced in version 1.4)

Searches for style having all properties of the **BaseStyle**-th style, but Charset⁷⁰⁰=**Charset**.

BaseStyle is an index in Items⁶⁷⁷.

The following properties are ignored when comparing styles:

- Standard⁶⁹⁵;
- StyleName⁶⁹⁶ (if RichViewCompareStyleNames¹⁰⁴⁴=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle⁶⁸¹;
- TFontInfo.IsEqual⁷¹¹.

Searches for style with the specified colors.

```
function FindStyleWithColor(BaseStyle: Integer;  
    Color, BackColor: TRVColor996): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but Color⁷⁰¹=**Color** and BackColor⁶⁹⁸=**BackColor**.

BaseStyle is an index in Items⁶⁷⁷.

The following properties are ignored when comparing styles:

- Standard⁶⁹⁵;
- StyleName⁶⁹⁶ (if RichViewCompareStyleNames¹⁰⁴⁴=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle⁶⁸¹;
- TFontInfo.IsEqual⁷¹¹.

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 Items⁶⁷⁷.

The following properties are ignored when comparing styles:

- Standard⁶⁹⁵;

- `StyleName`⁽⁶⁹⁶⁾ (if `RichViewCompareStyleNames`⁽¹⁰⁴⁴⁾=`False` (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other `TFontInfos.Find***` methods; the most universal method is `FindSuchStyle`⁽⁶⁸¹⁾;
- `TFontInfo.IsEqual`⁽⁷¹¹⁾.

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`⁽⁷⁰¹⁾=**FontName**.

BaseStyle is an index in `Items`⁽⁶⁷⁷⁾.

The following properties are ignored when comparing styles:

- `Standard`⁽⁶⁹⁵⁾,
- `StyleName`⁽⁶⁹⁶⁾ (if `RichViewCompareStyleNames`⁽¹⁰⁴⁴⁾=`False` (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other `TFontInfos.Find***` methods; the most universal method is `FindSuchStyle`⁽⁶⁸¹⁾;
- `TFontInfo.IsEqual`⁽⁷¹¹⁾.

Searches for style having the specified font size.

```
function FindStyleWithFontSize(BaseStyle: Integer;
  Size: TRVFontSize(1009)): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but font `Size`⁽⁷⁰⁷⁾=**Size**.

This method does not return items having odd values of `SizeDouble`⁽⁷⁰⁷⁾.

BaseStyle is an index in `Items`⁽⁶⁷⁷⁾.

The following properties are ignored when comparing styles:

- `Standard`⁽⁶⁹⁵⁾,
- `StyleName`⁽⁶⁹⁶⁾ (if `RichViewCompareStyleNames`⁽¹⁰⁴⁴⁾=`False` (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other `TFontInfos.Find***` methods; the most universal method is `FindSuchStyle`⁽⁶⁸¹⁾;
- `TFontInfo.IsEqual`⁽⁷¹¹⁾.

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⁽⁶⁷⁷⁾.

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⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle⁽⁶⁸¹⁾;
- TFontInfo.IsEqual⁽⁷¹¹⁾.

Searches for a style.

```
function FindSuchStyle(BaseStyle: Integer; Style: TFontInfo(712);
    Mask: TRVFontInfoProperties(1008)): Integer;
```

```
function FindSuchStyleEx(BaseStyle: Integer; Style: TFontInfo(712);
    Mask: TRVFontInfoProperties(1008)): 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⁽⁶⁷⁷⁾.

Mask is a set of properties to compare (other properties are ignored). Use RVAllFontInfoProperties⁽¹⁰⁰⁸⁾ constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=*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⁽⁶⁹⁵⁾=*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⁽⁶⁶¹⁾;
- TFontInfo.IsEqual⁽⁷¹¹⁾;
- TParaInfos.FindSuchStyle⁽⁶⁸⁵⁾;
- TRVListInfos.FindSuchStyle⁽⁶⁸⁶⁾.

Demos:

- Demos*\Editors\Editor 2\
(see procedure TForm1.rveStyleConversion, rve.OnStyleConversion⁽⁵⁸⁵⁾ event)

6.2.1.4.2 TParaInfos

Collection of paragraph attributes (styles). Class of TRVStyle.ParaStyles⁽⁶⁴⁸⁾ property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
```

```
TParaInfos = class(TCustomRVInfos)
```

Hierarchy

TObject

TPersistent

TCollection

TCustomRVInfos

Main Properties

Items⁽⁶⁸³⁾ – array of TParaInfo⁽⁷¹⁸⁾.

InvalidItem⁽⁶⁸³⁾ is returned when accessing item which is out of index range.

Methods for Paragraph Style Search

- FindStyleWithAlignment⁽⁶⁸⁴⁾;
- FindSuchStyle, FindSuchStyleEx⁽⁶⁸⁵⁾.

See Also

- TFontInfos⁽⁶⁷⁵⁾ – class of collection of text attributes (styles);
- TRVListInfos⁽⁶⁸⁵⁾ – class of collection of list attributes (i.e. styles of bullets and numbering).

6.2.1.4.2.1 Properties**In TParaInfos**

InvalidItem⁽⁶⁸³⁾

Items⁽⁶⁸³⁾

Derived from TCollection

Count

This item is returned when accessing `Items` [ItemNo], if (ItemNo<0) or (ItemNo>=TParaInfos.Count)

```
property InvalidItem: TParaInfo;
```

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value

Default value has all properties `Items` [0], but (LeftIndent,SpaceBefore,RightIndent,SpaceAfter)=(1,1,1,1) and red border.

See also

- `TFontInfos.InvalidItem`.

Provides indexed access to the items in the collection of paragraph attributes (styles)

```
property Items[Index: Integer]: TParaInfo;
```

This is default property of `TParaInfos`, so you can write

```
MyRVStyle.ParaStyles[i]
```

instead of

```
MyRVStyle.ParaStyles.Items[i]
```

Many methods of components inherited from `TCustomRichView` have `ParaNo` parameter, which is an index in `TParaInfos.Items` collection (more specifically, in `TCustomRichView.Style.ParaStyles.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.

6.2.1.4.2.2 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(718) ;
```

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⁽⁶⁸³⁾ array.

Return value:

Created collection item.

See also:

- TParaInfo.Create⁽⁷²⁷⁾.

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⁽⁶⁹⁶⁾) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.ParaStyles(648));
```

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⁽⁶⁴⁸⁾ 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⁽⁷¹⁹⁾=**Alignment**.

BaseStyle is an index in Items⁽⁶⁸³⁾.

The following properties are ignored when comparing styles:

- Standard⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=*False* (default)).

Return value:

Index of style or -1, if not found.

See also methods:

- FindSuchStyle⁽⁶⁸⁵⁾.

Searches for a style having properties like **Style**.

```
function FindSuchStyle(BaseStyle: Integer; Style: TParaInfo718;
  Mask: TRVParaInfoProperties1017): Integer;
function FindSuchStyleEx(BaseStyle: Integer; Style: TParaInfo718;
  Mask: TRVParaInfoProperties1017): Integer;
```

(Introduced in version 1.5 and version 14)

The **BaseStyle**-th style is checked first. **BaseStyle** is an index in Items⁶⁸³.

Mask is a set of properties to compare (other properties are ignored). Use RVAllParaInfoProperties¹⁰¹⁷ constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard⁶⁹⁵;
- StyleName⁶⁹⁶ (if RichViewCompareStyleNames¹⁰⁴⁴=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 Standard⁶⁹⁵=False, returns the index of this new item.

Return value:

Index of the found style; **FindSuchStyle** returns -1 if not found.

See also methods:

- FindStyleWithAlignment⁶⁸⁴;
- TRVStyle.FindParaStyle⁶⁶¹;
- TParaInfo.IsEqual⁷²⁷;
- TFontInfos.FindSuchStyle⁶⁶¹;
- TRVListInfos.FindSuchStyle⁶⁸⁶.

Demos:

- Demos*\Editors\Editor 2\
(see procedure TForm1.rveParaStyleConversion, rve.OnParaStyleConversion⁵⁸² event)

6.2.1.4.3 TRVListInfos

Collection of paragraph list attributes (bullets and numbering). Class of TRVStyle.ListStyles⁶⁴⁶ property.

Unit RVStyle.

Syntax

```
TCustomRVInfos = class(TCollection)
TRVListInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items⁶⁸⁶ – array of TRVListInfo⁷³¹.

Main Methods

- FindSuchStyle⁶⁸⁶
- FindStyleWithLevels⁶⁸⁷

See Also

- TFontInfos⁶⁷⁵ – class of collection of text attributes (styles);
- TParainfos⁶⁸² – class of collection of paragraph attributes (styles);
- List markers item type¹⁷⁴.

6.2.1.4.3.1 Properties

In TRVListInfos

Items⁶⁸⁶

Derived from TCollection

Count

Provides indexed access to items in the collection of paragraph list attributes (bullets and numbering)

```
property Items[Index:Integer] : TRVListInfo731;
```

This is default property of TRVListInfos, so you can write

```
MyRVStyle.ListStyles646[i]
```

instead of

```
MyRVStyle.ListStyles.Items[i]
```

6.2.1.4.3.2 Methods

In TRVListInfos

Add⁶⁸⁷

FindStyleWithLevels⁶⁸⁷

FindSuchStyle⁶⁸⁶

Derived from TCustomRVInfos

AssignTo⁶⁸⁷

You can also use methods derived from TCollection

Searches for the style. Adds if not found (optionally)

```
function FindSuchStyle(Style: TRVListInfo731;  
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**⁽⁶⁹⁵⁾ property = *False*.

The following properties are ignored when comparing styles:

- **Standard**⁽⁶⁹⁵⁾.
- **StyleName**⁽⁶⁹⁶⁾ (if **RichViewCompareStyleNames**⁽¹⁰⁴⁴⁾=*False* (default)).

Return value:

Index of style, or -1 if not found (and not added).

See also:

- **FindStyleWithLevels**⁽⁶⁸⁷⁾;
- **TFontInfos.FindSuchStyle**⁽⁶⁸¹⁾;
- **TParaInfos.FindSuchStyle**⁽⁶⁸⁵⁾.

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**⁽⁷³²⁾ and **StyleNameForAdding** to its **StyleName**⁽⁶⁹⁶⁾. The created style has **Standard**⁽⁶⁹⁵⁾ property = *False*.

Return value:

Index of style, or -1 if not found (and not added).

See also:

- **FindSuchStyle**⁽⁶⁸⁶⁾.

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⁽⁶⁹⁶⁾) to TStrings.

Do not call AssignTo directly, this method is called from **Dest.Assign** method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.ListStyles(646));
```

Adds a new style to the collection of paragraph list styles (bullets and numbering)

```
function Add: TRVListInfo(731);
```

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**⁽⁶⁸⁸⁾ array.

Add returns the new collection item.

6.2.1.4.4 TRVStyleTemplateCollection

Collection of "real" styles of text and paragraphs. Class of TRVStyle.StyleTemplates⁽⁶⁵²⁾ property.

Unit RVStyle.

Syntax

```
TRVStyleTemplateCollection = class(TCollection)
```

(introduced in version 14)

Hierarchy

TObject

TPersistent

TCollection

Main Properties

- Items⁽⁶⁸⁹⁾ – array of TRVStyleTemplate⁽⁷³³⁾.
- DefStyleName⁽⁶⁸⁸⁾ – a format string used to generate names of new items.

Main Methods

- AssignStyleTemplates⁽⁶⁹⁰⁾ assigns another collection to this collection;
- FindById, FindItemById⁽⁶⁹¹⁾ search for items by their identifiers;
- FindByName, FindItemByName⁽⁶⁹¹⁾ search for items by their names;
- LoadFromRVST, SaveToRVST⁽⁶⁹²⁾ save and load the collection to a file;
- LoadFromStreamRVST, SaveToStreamRVST⁽⁶⁹²⁾ save and load the collection to a stream (Delphi 10.3+);
- ResetNameCounter⁽⁶⁹³⁾ resets a counter used to generate names for new items.

See also

- Styles and style templates⁽⁹⁸⁾.

6.2.1.4.4.1 Properties

In TRVStyleTemplateCollection

- DefStyleName⁽⁶⁸⁸⁾
- ▶ ForbiddenIds⁽⁶⁸⁹⁾
- ▶ Items⁽⁶⁸⁹⁾
- ▶ NormalStyleTemplate⁽⁶⁸⁹⁾

Derived from TCollection

Count

Specifies the name for new style templates.

```
property DefStyleName: TRVUnicodeString(1032);
```

(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⁽⁷³⁵⁾ of new items in this collection.

Default value:

'Style %d'

See also:

- ResetNameCounter⁶⁹³

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`⁵⁰⁰. When the user deletes a style template, add its `Id`⁷³⁴ 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]: TRVStyleTemplate733;
```

This is default property of `TRVStyleTemplateCollection`, so you can write

```
MyRVStyle.StyleTemplates652[i]
```

instead of

```
MyRVStyle.StyleTemplates.Items[i]
```

Returns the item having `Name`⁷³⁵='Normal'.

```
property NormalStyleTemplate: TRVStyleTemplate733;
```

If such an item does not exist in `Items`⁶⁸⁹, the property returns *nil*.

6.2.1.4.4.2 Methods

In `TRVStyleTemplateCollection`

```
AssignStyleTemplates690  
AssignToStrings690  
ClearParaFormat690  
ClearTextFormat690  
FindById691  
FindByName691  
FindItemById691  
FindItemByName691  
LoadFromRVST692  
LoadFromStreamRVST692 (D10.3+)  
ResetNameCounter693  
SaveToRVST692  
SaveToStreamRVST692 (D10.3+)
```

Copies items from **Source** to this collection.

```
procedure AssignStyleTemplates(Source: TRVStyleTemplateCollection;
CopyIds: Boolean);
```

If **CopyIds=True**, Id⁽⁷³⁴⁾ properties of **Source.Items**⁽⁶⁸⁹⁾ are copied to new items.

If **CopyIds=False**, the method generates new Id for new items, retaining NextId⁽⁷³⁶⁾ and ParentId⁽⁷³⁷⁾ relations.

Note:

StyleTemplates1.Assign(StyleTemplates2) is equivalent to
StyleTemplates1.AssignStyleTemplates(StyleTemplates2, False).

See also:

- TCustomRichViewEdit.ChangeStyleTemplates⁽⁵⁰⁰⁾.

Assign style templates to a string list.

type

```
TRVStyleTemplateKinds = set of TRVStyleTemplateKind(735);
```

```
procedure AssignToStrings(Strings: TStrings; AssignIds: Boolean;
Kinds: TRVStyleTemplateKinds; ExcludeThisStyle: TRVStyleTemplate(733);
OnlyPotentialParents: Boolean);
```

Parameters:

Strings – a destination string list. Names⁽⁷³⁵⁾ of style templates are added as items to this string list.

If **AssignIds=True**, the method saves Ids⁽⁷³⁴⁾ of style templates as Objects of **Strings** (type-casted to TObject).

Only style templates having Kinds⁽⁷³⁵⁾ 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⁽⁷³⁵⁾ 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(696));
```

```
procedure ClearParaFormat(AParaStyle: TCustomRVParaInfo(718));
```

ClearTextFormat clears formatting from **ATextStyle**. The method resets all its properties to default values, then:

- if **ATextStyle** is TFontInfo⁽⁷¹²⁾, and **ATextStyle.Jump**⁽⁷¹⁴⁾=True, and "Hyperlink" style template exists in Items⁽⁶⁸⁹⁾, 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 Items⁶⁸⁹, the method applies it to **AParaStyle**;
- otherwise, the method assigns **AParaStyle.StyleTemplateId** = -1.

Searches for the item having the specified identifier⁷³⁴ **Id**.

```
function FindById(Id: TRVStyleTemplateId): Integer;
```

```
function FindItemById(Id: TRVStyleTemplateId): TRVStyleTemplate;
```

FindById returns the index of the found item in Items⁶⁸⁹, or -1 if not found.

FindItemById returns the the found item, or *nil* if not found.

See also:

- FindByName, FindItemByName⁶⁹¹

Searches for the item having the specified name⁷³⁵ **Name**

```
function FindByName(const Name: TRVStyleTemplateName1027): Integer;
```

```
function FindItemByName(const Name: TRVStyleTemplateName1027):  
TRVStyleTemplate;
```

FindByName returns the index of the found item in Items⁶⁸⁹, or -1 if not found.

FindItemByName returns the the found item, or *nil* if not found.

See also:

- FindById, FindItemById⁶⁹¹

Applies styles from the file to the collection.

```
function InsertFromRVST(const FileName: TRVUnicodeString1032;  
Units: TRVStyleUnits1027): Boolean;
```

Files of "RichView Styles" format contain style templates. Normally they have "rvst" extension. They can be created by SaveToRVST⁶⁹² 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: TRVStyleTemplateCollection688;  
begin  
  if not OpenDialog1.Execute then  
    exit;  
  StyleTemplates := TRVStyleTemplateCollection688.Create(nil);  
  try  
    StyleTemplates.AssignStyleTemplates690(rvs.StyleTemplates, True);  
    if not StyleTemplates.InsertFromRVST(OpenDialog1.FileName,
```

```

RichViewEdit1.Style(253).Units(655) ) then
  exit;
RichViewEdit1.ChangeStyleTemplates(500) (StyleTemplates);
finally
  StyleTemplates.Free;
end;
end;

```

The methods load and save style templates to files.

```

function LoadFromRVST(const FileName: TRVUnicodeString(1032);
  Units: TRVStyleUnits(1027);
  ARVStyle: TRVStyle(630) = nil): Boolean;
function SaveToRVST(const FileName: TRVUnicodeString(1032);
  Units: TRVStyleUnits(1027)): 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⁽⁶⁵²⁾, this is RVStyle.Units⁽⁶⁵⁵⁾;
- 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⁽⁶³⁰⁾ 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⁽⁶⁵²⁾, clear RVStyle.TextStyles⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾ (because the old items may contain references⁽⁶⁹⁶⁾ to previous style templates). Then you can add default text and paragraph styles (see the example⁽⁷³⁸⁾).

Example:

Demos*\Editors\StyleTemplates\ demo loads "default.rvst" in "File | New" command.

See also

- InsertFromRVST⁽⁶⁹¹⁾
- LoadFromStreamRVST, SaveToStreamRVST⁽⁶⁹²⁾

The methods load and save style templates to streams.

```

function SaveToStreamRVST(Stream: TStream;
  Units: TRVStyleUnits(1027)): Boolean;
function LoadFromStreamRVST(Stream: TStream;
  Units: TRVStyleUnits(1027); ARVStyle: TRVStyle(630) = nil): Boolean;

```

(introduced in version 19; for Delphi and C++Builder 10.3+)

These methods are analogs `SaveToRVST` and `LoadFromRVST`⁶⁹², but they save/load to/from **Stream** instead of a file.

Resets the counter used to generate unique names for new Items⁶⁸⁹.

```
procedure ResetNameCounter;
```

Names are generated using `DefStyleName`⁶⁸⁸ 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.

6.2.1.5 Classes of properties - Styles

Classes defining formatting

Main classes:

- `TFontInfo`⁷¹² (inherited from `TCustomRVFontInfo`⁶⁹⁶) – a type of items in `TRVStyle`⁶³⁰.
`.TextStyles`⁶⁵⁴ property collection, contains character properties.
- `TParaInfo`⁷²⁷ (inherited from `TCustomRVParaInfo`⁷¹⁸) – a type of items in `TRVStyle.ParaStyles`⁶⁴⁸
property collection, contains paragraph properties.
- `TRVListInfo`⁷³¹ – a type of items in `TRVStyle.ListStyles`⁶⁴⁶ property collection, contains properties of paragraph lists (bullets and numbering)

Ancestor classes:

- `TCustomRVInfo`, `TCustomRVTextOrParaInfo`⁶⁹³

"Real styles"

- `TRVStyleTemplate`⁷³³ – a type of items in `TRVStyle.StyleTemplates`⁶⁵² property, represent a named style.

6.2.1.5.1 TCustomRVInfo, TCustomRVTextOrParaInfo

These are ancestor classes for all `TRichView` attributes (styles): text attributes, paragraph attributes, list attributes.

Unit `RVStyle`.

Syntax

```
TCustomRVInfo = class(TCollectionItem)
TCustomRVFontOrParaInfo = class(TCustomRVInfo)
```

Hierarchy

TObject

TPersistent

TCollectionItem

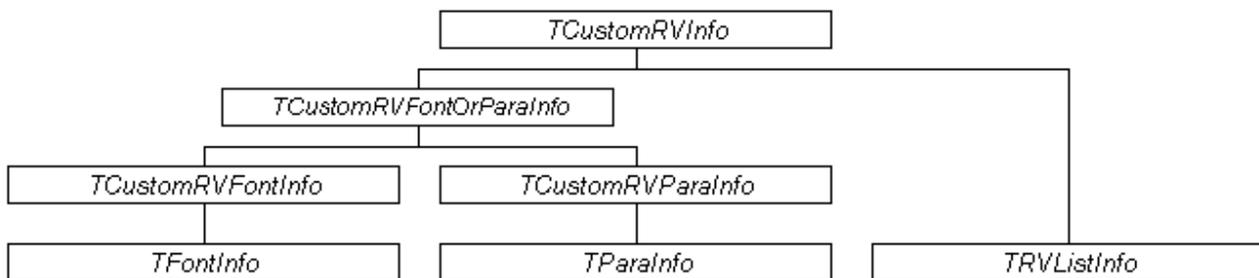
Using

These classes are not used directly. Classes inherited from them are used instead.

The following classes are inherited from TCustomRVInfo (all defined in RVStyle unit).

Class	Meaning
TCustomRVFontInfo ⁽⁶⁹⁶⁾	This class introduces the most of properties related to text attributes. Ancestor of TFontInfo.
TCustomRVParaInfo ⁽⁷¹⁸⁾	This class introduces the most of properties related to paragraph attributes. Ancestor of TParaInfo.
TFontInfo ⁽⁷¹²⁾	Text style, defines text attributes. Class of items in the collection TRVStyle ⁽⁶³⁰⁾ .TextStyles ⁽⁶⁵⁴⁾ .
TParaInfo ⁽⁷²⁷⁾	Paragraph style, defines paragraph attributes. Class of items in the collection TRVStyle ⁽⁶³⁰⁾ .ParaStyles ⁽⁶⁴⁸⁾ .
TRVListInfo ⁽⁷³¹⁾	List style (i.e. style of paragraphs' bullets & numbering). It has Levels ⁽⁷³²⁾ collection defining attributes of paragraph list markers ⁽¹⁷⁴⁾ . Class of items in the collection TRVStyle ⁽⁶³⁰⁾ .ListStyles ⁽⁶⁴⁶⁾ .

The class hierarchy is explained on the diagram below.



Properties

TCustomRVInfo introduces the following properties:

- Standard ⁽⁶⁹⁵⁾.
- StyleName ⁽⁶⁹⁶⁾.

TCustomRVFontOrParaInfo introduces the following properties:

- StyleTemplateId ⁽⁶⁹⁶⁾.

6.2.1.5.1.1 Properties

In TCustomRVFontOrParaInfo

- StyleTemplateId ⁽⁶⁹⁶⁾

In TCustomRVInfo

- Standard⁶⁹⁵
- StyleName⁶⁹⁶

Identifies standard (essential) styles

```
property Standard: Boolean;
```

(introduced in version 1.6)

Description

Standard styles are not deleted by DeleteUnusedStyles²⁸⁶.

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.FindTextStyle⁶⁶¹, TRVStyle.FindParaStyle⁶⁶¹, TFontInfos.FindSuchStyleEx⁶⁸¹, TParaInfos.FindSuchStyleEx⁶⁸⁵ are not standard.

Styles which are added automatically while loading from RVF preserve value of this property.

Usage without StyleTemplates⁶⁵²

If StyleTemplates are not used²⁵⁶, 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 styles²¹⁸, 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 StyleTemplates⁶⁵²

If StyleTemplates are used, it's recommended to have only one standard paragraph style initially, linked⁶⁹⁶ to NormalStyleTemplate⁶⁸⁹, and only one standard text style not linked to any style template (but its ParaStyleTemplateId⁷⁰⁵ should point to NormalStyleTemplate⁶⁸⁹ as well). All other styles will be added on editing.

Default value:

True

See also:

- TFontInfo⁷¹² – class of text attributes (style);
- TParaInfo⁷¹⁸ – class of paragraph attributes (style);
- TRVListInfo⁷³¹ – class of paragraph list attributes (style);
- RichViewResetStandardFlag¹⁰⁴⁷ typed constant.

Name of this style.

```
property StyleName: TRVUnicodeString1032;
```

(changed in version 18)

This name is used:

- when displaying a design-time editor for collections of styles;
- when assigning text styles to TStrings⁶⁷⁸.
- when saving RTF without using StyleTemplates²⁵⁶ and with *rvtfSaveStyleSheet*²⁴⁵ option;

This property is ignored when comparing and searching styles (see also *RichViewCompareStyleNames*¹⁰⁴⁴ typed constant).

Names of styles do not need to be unique in the collection (unless you save RVF with *rvfoUseStyleNames*²⁴⁷ option).

This property makes sense if you use a predefined set of styles²¹⁸. If you add styles dynamically, you can ignore this property.

Links this style to a style template.

```
property StyleTemplateId: TRVStyleTemplateId1027;
```

(introduced in version 14)

This property is used only if style templates are used²⁵⁶.

TCustomRVFontInfo⁶⁹⁶ may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkText*.

TCustomRVParaInfo⁷¹⁸ may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkPara*.

See also:

- TCustomRVFontInfo.ParaStyleTemplateId⁷⁰⁵

6.2.1.5.2 TCustomRVFontInfo

This is an ancestor class for TFontInfo⁷¹² (class of item in collection of text styles⁶⁵⁴).

TCustomRVFontInfo introduces the most of properties of text.

Unit RVStyle.

Syntax

```
TCustomRVFontInfo = class(TCustomRVFontOrParaInfo693)
```

Hierarchy

TObject

TPersistent

TCollectionItem

TCustomRVFontOrParaInfo⁶⁹³

TCustomRVInfo⁽⁶⁹³⁾

Properties

Font attributes (similar to properties of TFont):

- `FontName`⁽⁷⁰¹⁾ – font name;
- `Size`⁽⁷⁰⁷⁾ and `SizeDouble`⁽⁷⁰⁷⁾ – font size;
- `Color`⁽⁷⁰¹⁾ – text color;
- `Style`⁽⁷⁰⁸⁾ – text styles (bold, italic, underline, strikethrough);
- `Charset`⁽⁷⁰⁰⁾ – font character set (for non-Unicode text styles).

Additional properties affecting text appearance and layout:

- `BackColor`⁽⁶⁹⁸⁾ – text background color;
- `UnderlineColor`⁽⁷⁰⁹⁾ – underline color;
- `UnderlineType`⁽⁷⁰⁹⁾ – underline style (normal, double, dotted, etc.);
- `StyleEx`⁽⁷⁰⁸⁾ – additional text styles (overline, all-caps);
- `SubSuperScriptType`⁽⁷⁰⁸⁾ – normal text/subscript/superscript;
- `VShift`⁽⁷¹⁰⁾ moves text up or down from its base line;
- `CharScale`⁽⁶⁹⁹⁾ – horizontal character scaling;
- `CharSpacing`⁽⁷⁰⁰⁾ – spacing between characters;
- `BiDiMode`⁽⁶⁹⁹⁾ – default text direction.
- `EmptyWidth`⁽⁷⁰¹⁾ – width for empty text item

Hypertext properties:

- `HoverColor`⁽⁷⁰²⁾ – text highlight color;
- `HoverBackColor`⁽⁷⁰²⁾ – text background highlight color;
- `HoverUnderlineColor`⁽⁷⁰³⁾ – underline highlight color;
- `HoverEffects`⁽⁷⁰³⁾ – additional highlight effects (underline);
- `JumpCursor`⁽⁷⁰⁴⁾ – mouse cursor for this text.

Properties related to editing in TCustomRichViewEdit⁽⁴⁶¹⁾:

- `Protection`⁽⁷⁰⁵⁾ – text protection options;
- `EmptyWidth`⁽⁷⁰¹⁾ – width for empty text item, affects editing.

Properties related to text export:

- `Options`⁽⁷⁰⁴⁾ allows to create RTF and HTML codes.

Methods

`IsEqual`⁽⁷¹¹⁾ compares this text style with another one.

You can use `Assign`⁽⁷¹¹⁾ method to assign:

- `TCustomRVFontInfo` to `TCustomRVFontInfo`;
- `TFont` to `TCustomRVFontInfo`;
- `TCustomRVFontInfo` to `TFont`.

See Also

Classes:

- `TFontInfo`⁽⁷¹²⁾ – class inherited from this class, text attributes (style);
- `TFontInfos`⁽⁶⁷⁵⁾ – collection of text attributes (styles) (`TFontInfo`⁽⁷¹²⁾).

6.2.1.5.2.1 Properties

In TCustomRVFontInfo

- BackColor⁽⁶⁹⁸⁾
- BiDiMode⁽⁶⁹⁹⁾
- CharScale⁽⁶⁹⁹⁾
- Charset⁽⁷⁰⁰⁾
- CharSpacing⁽⁷⁰⁰⁾
- Color⁽⁷⁰¹⁾
- EmptyWidth⁽⁷⁰¹⁾
- FontName⁽⁷⁰¹⁾
- HoverBackColor⁽⁷⁰²⁾
- HoverColor⁽⁷⁰²⁾
- HoverEffects⁽⁷⁰³⁾
- HoverUnderlineColor⁽⁷⁰³⁾
- JumpCursor⁽⁷⁰⁴⁾
- Options⁽⁷⁰⁴⁾
- ParaStyleTemplateId⁽⁷⁰⁵⁾
- Protection⁽⁷⁰⁵⁾
- Size⁽⁷⁰⁷⁾
- SizeDouble⁽⁷⁰⁷⁾
- Style⁽⁷⁰⁸⁾
- StyleEx⁽⁷⁰⁸⁾
- SubSuperScriptType⁽⁷⁰⁸⁾
- UnderlineColor⁽⁷⁰⁹⁾
- UnderlineType⁽⁷⁰⁹⁾
- VShift⁽⁷¹⁰⁾

Derived from TCustomRVFontOrParaInfo⁽⁶⁹³⁾

- StyleTemplateId⁽⁶⁹⁶⁾

Derived from TCustomRVInfo⁽⁶⁹³⁾

- Standard⁽⁶⁹⁵⁾
- StyleName⁽⁶⁹⁶⁾

Background color for text of this style

```
property BackColor: TRVColor(996);
```

Set BackColor=*rvclNone*⁽¹⁰³⁸⁾ for transparent text background.

Default value:

rvclNone⁽¹⁰³⁸⁾

See also:

- HoverBackColor⁽⁷⁰²⁾;
- Color⁽⁷⁰¹⁾;
- HoverColor⁽⁷⁰²⁾.

See also properties of TRVStyle:

- `TextBackgroundKind`⁶⁵³.

Specifies default direction for the text of this style.

property `BiDiMode`: `TRVBiDiMode`⁹⁹³;

(introduced in version 1.6)

Value	Meaning
<code>rvbdUnspecified</code>	Text direction is inherited from higher level, i.e. from paragraph (see <code>BiDiMode</code> ⁷²¹ of paragraph style)
<code>rvbdLeftToRight</code>	Default text reading order is left-to-right
<code>rvbdRightToLeft</code>	Default text reading order is right-to-left

Default value:

`rvbdUnspecified`

See also:

- Bidirectional text in `TRichView`¹³².

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.

100%

text

150%

text

80%

text

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 drawing⁶⁷¹. 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(1008);
```

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_CHARSET 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(1027);
```

(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⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns this text style.

Example:

This example shows text formatted with CharSpacing = 0, 5, and -3 pixels.

0

text

5

text

-3

text

CharSpacing
Example

Note: if TextEngine⁽⁶⁵⁴⁾ is Windows API, all expanded characters are aligned to the left side (space is added to the right side of the characters).

if TextEngine⁽⁶⁵⁴⁾ 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⁽¹⁰³⁸⁾

See also:

- *BackColor*⁽⁶⁹⁸⁾;
- *HoverColor*⁽⁷⁰²⁾;
- *HoverBackColor*⁽⁷⁰²⁾.

Specifies the width of a text item if its text is empty.

```
property EmptyWidth: TRVStyleLength(1027);
```

(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⁽⁶⁹⁸⁾ for them.

For all normal text items, selection⁽¹⁰⁷⁾/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*⁽⁷⁰⁵⁾.

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: TRVColor⁹⁹⁶;

(introduced in version 1.2)

If this style is a hypertext style (Jump⁷¹⁴=True), background of text of this style can be highlighted with this color when user moves the mouse pointer above it. Set HoverBackColor=BackColor⁶⁹⁸ to disable effect.

By default, RichViewEdit⁴⁶¹ also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions⁴⁷⁵ property.

This property has no effect for non-hypertext styles.

Default value:

*rvclNone*¹⁰³⁸

See also properties:

- BackColor⁶⁹⁸;
- HoverColor⁷⁰²;
- Jump⁷¹⁴;
- JumpCursor⁷⁰⁴;
- HoverEffects⁷⁰³;
- HoverUnderlineColor⁷⁰³.

See also properties of TRVStyle:

- HoverColor⁶⁴¹;
- FullRedraw⁶⁴⁰;
- TextBackgroundKind⁶⁵³.

Specifies highlight color for hypertext links of this style.

property HoverColor: TRVColor⁹⁹⁶;

If this style is a hypertext style (Jump⁷¹⁴=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*¹⁰³⁸, TRVStyle.HoverColor⁶⁴¹ is used. If it is, in its order, equal to *rvclNone*¹⁰³⁸, no hover color effect is used.

By default, RichViewEdit⁴⁶¹ also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions⁴⁷⁵ property.

This property has no effect for non-hypertext styles.

Default value:

*rvclNone*¹⁰³⁸

See also properties:

- Color⁷⁰¹;
- HoverBackColor⁷⁰²;
- Jump⁷¹⁴;
- JumpCursor⁷⁰⁴;

- `HoverEffects`⁽⁷⁰³⁾;
- `HoverUnderlineColor`⁽⁷⁰³⁾.

See also properties of TRVStyle:

- `HoverColor`⁽⁶⁴¹⁾;
- `FullRedraw`⁽⁶⁴⁰⁾.

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`⁽⁷¹⁴⁾ = `True`), these effects are applied to text of this style when user moves the mouse pointer above it.

By default, `RichViewEdit`⁽⁴⁶¹⁾ also applies these effects to hyperlinks when the caret is moved inside them. This effect can be disabled by including `rvNoCaretHighlightJumps` in `RichViewEdit.EditorOptions`⁽⁴⁷⁵⁾ property.

This property has no effect for non-hypertext styles.

Effect	Meaning
<code>rvheUnderline</code>	Text is underlined.

This underline may have a custom color, see `HoverUnderlineColor`⁽⁷⁰³⁾ property.

Default value:

[]

See also properties:

- `Jump`⁽⁷¹⁴⁾;
- `HoverColor`⁽⁷⁰²⁾;
- `HoverBackColor`⁽⁷⁰²⁾;
- `JumpCursor`⁽⁷⁰⁴⁾.

See also properties of TRVStyle:

- `FullRedraw`⁽⁶⁴⁰⁾.

Specifies underline highlight color for hypertext links of this style.

property `UnderlineColor`: TRVColor⁽⁹⁹⁶⁾;

(introduced in version 10)

Underline is displayed only if `fsUnderline` is included in `Style`⁽⁷⁰⁸⁾, or `rvheUnderline` is included `HoverEffects`⁽⁷⁰³⁾.

If this style is a hypertext style (`Jump`⁽⁷¹⁴⁾ = `True`), underline of text of this style can be highlighted with this color when user moves the mouse pointer above it.

If this property is equal to *rvclNone*⁽¹⁰³⁸⁾, *UnderlineColor*⁽⁷⁰⁹⁾ is used. If it is, in its order, equal to *rvclNone*⁽¹⁰³⁸⁾, *HoverColor*⁽⁷⁰²⁾ is used. If it is, in its order, equal to *rvclNone*⁽¹⁰³⁸⁾, *Color*⁽⁷⁰¹⁾ 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*⁽²³⁷⁾.*HTMLSavingType*⁽⁴³³⁾ 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⁽¹⁰³⁸⁾

See also:

- *UnderlineType*⁽⁷⁰⁹⁾.

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*⁽⁷¹⁴⁾=*True*).

Default value:

crJump

See also properties of TRVStyle:

- *JumpCursor*⁽⁶⁴³⁾.

See also:

- Cursors used by components of *TRichView* family⁽¹⁴²⁾.

Options for exporting text of this style to RTF and HTML.

type

```
TRVTextOption = (rvteoHTMLCode, rvteoRTFCode, rvteoDocXCode,
  rvteoDocXInRunCode, rvteoHidden);
TRVTextOptions = set of TRVTextOption;
```

```
property NextStyleNo: Integer;
```

(introduced in version 1.6)

Option	Meaning
<i>rvteoRTFCode</i>	If set, '{', '}', '\ ' are saved in RTF as they are. Example: set this option, write text of this style: <code>\animtext1 Viva Las Vegas'</code> , copy to RTF and paste in MS Word.
<i>rvteoHTMLCode</i>	If set, '&', '<', '>', and multiple spaces are saved in HTML as they are.

Option	Meaning
	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.
<i>rvteoDocXCode</i>	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.
<i>rvteoDocXInRunCode</i>	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>).
<i>rvteoHidden</i>	If set, this text is hidden. Hidden text is displayed only if <i>rvShowHiddenText</i> is included in RichView.Options ⁽²⁴⁰⁾

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 ⁽³⁴³⁾;
- TCustomRichView.SaveDocX ⁽³³³⁾;
- About HTML export ⁽¹²⁷⁾.

Links this text style to the style template of paragraph.

property ParaStyleTemplateId: TRVStyleTemplateId ⁽¹⁰²⁷⁾;

(introduced in version 14)

This property is used only if style templates are used ⁽²⁵⁶⁾.

This property is redundant: the style template of paragraph is linked using TParaInfo ⁽⁷²⁷⁾.StyleTemplateId ⁽⁶⁹⁶⁾. 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 ⁽⁶⁵⁴⁾, 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⁽⁴⁶¹⁾. Text of this style has special behavior when editing.

General Protection Options

Option	Meaning
<i>rvprStyleProtect</i>	Protects text from changing style with ApplyTextStyle ⁽⁴⁹⁴⁾ , ApplyTextStyleTemplate ⁽⁴⁹⁴⁾ , ApplyStyleTemplate ⁽⁴⁹³⁾ . This protection is ignored by ApplyStyleConversion ⁽⁴⁹²⁾ , but you can take it into account manually in OnStyleConversion ⁽⁵⁸⁵⁾ event.
<i>rvprStyleSplitProtect</i>	Protects from applying text style to the part of text item. ApplyTextStyle ⁽⁴⁹⁴⁾ and ApplyStyleConversion ⁽⁴⁹²⁾ methods apply new style to the whole item having this protection, even if it is selected only partially.
<i>rvprModifyProtect</i>	Protects text items of this style from modification, but does not protect from deleting as a whole. Deletion depends on <i>rvFastDeleteProtectedText</i> option in TRichViewEdit.EditorOptions ⁽⁴⁷⁵⁾ .
<i>rvprDeleteProtect</i>	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 ⁽⁷⁰¹⁾ > 0, all characters can be deleted, but the item is not deleted. See also: <i>rvDeleteProtect</i> item property ⁽¹⁰⁰⁰⁾ .
<i>rvprConcatProtect</i>	Protects texts items from concatenation with adjacent item of the same style.
<i>rvprRVFInsertProtect</i>	If set, items of this style will not be inserted when inserting RVF. Affects methods introduced in editor only (InsertRV***Ed , pasting).
<i>rvprDoNotAutoSwitch</i>	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 ⁽⁴⁷⁴⁾ property), the editor switches back to another (previously used) style after the first performed operation.

Option	Meaning
<i>rvprParaStartProtect</i>	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
<i>rvprSticking</i>	Disallows insertion between two text items having this protection (allows to stick together several text items of different text styles).
<i>rvprSticking2</i>	The same.
<i>rvprSticking3</i>	The same.
<i>rvprStickToTop</i>	Disallows insertion before the text of this style, if this text is a first document item.
<i>rvprStickToBottom</i>	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` ⁽⁵⁷³⁾ event.

Note:

`TRichView.FontInfo.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 ⁽⁷²³⁾ of paragraph style;
- `EmptyWidth` ⁽⁷⁰¹⁾ property

Demo projects:

- `Demos*\Assorted\Fields\`

The properties specify the font height.

```
property Size: TRVFontSize(1009) ;
property SizeDouble: TRVFontSize(1009) ;
```

(*SizeDouble* is introduced in version 15)

Size specifies the font size in points ⁽¹³⁹⁾.

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:

- StyleEx⁷⁰⁸;
- HoverEffects⁷⁰³.

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
<i>rvfsOverline</i>	Vertical line above the text
<i>rvfsAllCaps</i>	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` ⁽⁷⁰⁷⁾.

Default value:

`rvssNormal`

See also properties:

- `VShift` ⁽⁷¹⁰⁾.

Specifies underline color.

property `UnderlineColor: TRVColor` ⁽⁹⁹⁶⁾;

(introduced in version 10)

Underline is displayed only if `fsUnderline` is included in `Style` ⁽⁷⁰⁸⁾ (or `rvheUnderline` is included `HoverEffects` ⁽⁷⁰³⁾, under the mouse pointer).

If `UnderlineColor=rvclNone` ⁽¹⁰³⁸⁾, text color is used for underline (`Color` ⁽⁷⁰¹⁾ in normal state, `HoverColor` ⁽⁷⁰²⁾ for hyperlinks under the mouse pointer).

Values other than `rvclNone` ⁽¹⁰³⁸⁾ define underline color. `HoverUnderlineColor` ⁽⁷⁰³⁾ 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` ⁽²³⁷⁾.`HTMLSavingType` ⁽⁴³³⁾ 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` ⁽¹⁰³⁸⁾

See also:

- `UnderlineType` ⁽⁷⁰⁹⁾.

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` ⁽⁷⁰⁸⁾ (or `rvheUnderline` is included `HoverEffects` ⁽⁷⁰³⁾, under the mouse pointer).

By default, underline color is defined in Color⁽⁷⁰¹⁾ property, but it can be altered by UnderlineColor⁽⁷⁰⁹⁾ property (see also HoverColor⁽⁷⁰²⁾/HoverUnderlineColor⁽⁷⁰³⁾ for hyperlink colors under the mouse pointer).

Sample:

rvutNormal, rvutThick, rvutDouble,
rvutDotted, rvutThickDotted,
rvutDashed, rvutThickDashed,
rvutLongDashed, rvutThickLongDashed,
rvutDashDotted, rvutThickDashDotted,
rvutDashDotDotted, rvutThickDashDotDotted

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⁽²³⁷⁾.HTMLSavingType⁽⁴³³⁾ 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⁽⁷⁰⁸⁾ (the effect is cumulative).

Default value:

0

See also:

- TRVExtraItemProperty⁽¹⁰⁰⁰⁾ type (*rvepVShift*)

6.2.1.5.2.2 Methods

In TCustomRVFontInfo

Assign⁽⁷¹¹⁾

AssignTo⁽⁷¹¹⁾

Create⁽⁷¹¹⁾

IsEqual⁽⁷¹¹⁾

Copies the contents of **Source** to this text style.

procedure Assign(Source: TPersistent); **override**;

Call Assign to copy properties of TFont, TCustomRVInfo⁽⁶⁹³⁾, or TCustomRVFontInfo to this text style.

See also:

- TFontInfo.Assign⁽⁷¹⁵⁾ (TFontInfo⁽⁷¹²⁾ 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(654)[0]);
```

A constructor, creates and initializes a new instance of TCustomRVFontInfo.

constructor Create(Collection: TCollection); **override**;

The constructor initializes:

```
// Properties inherited from TCustomRVInfo(693):
Standard(695)      := True;
StyleName(696)   := 'Font Style';
// Properties defined in TCustomRVFontInfo:
Size(707)        := 10;
FontName(701)    := 'Arial';
Color(701)       := clWindowText;
UnderlineColor(709) := clNone;
BackColor(698)   := clNone;
HoverColor(702)  := clNone;
HoverUnderlineColor(703) := clNone;
Charset(700)     := DEFAULT_CHARSET;
JumpCursor(704) := crDefault;
CharScale(699)  := 100;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a text style in code, you should assign Standard⁽⁶⁹⁵⁾ := *False*.

Compares this text style with **Value**.

function IsEqual(Value: TCustomRVFontInfo; IgnoreList: TRVFontInfoProperties⁽¹⁰⁰⁸⁾)

(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⁽¹⁰⁰⁸⁾.

The following properties are always ignored when comparing styles:

- Standard⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=False (default)).

See also:

- TFontInfo.IsEqual⁽⁷¹⁷⁾ (TFontInfo⁽⁷¹²⁾ is inherited from this class);
- TCustomRVParaInfo.IsEqual⁽⁷²⁷⁾.

6.2.1.5.3 TFontInfo

This class defines attributes of text items⁽¹⁶¹⁾ in TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ controls.

This is an item in the collection of TRVStyle⁽⁶³⁰⁾.TextStyles⁽⁶⁵⁴⁾. The class of the collection itself is TFontInfos⁽⁶⁷⁵⁾.

Unit RVStyle.

Syntax

```
TFontInfo = class(TCustomRVFontInfo(696))
```

Hierarchy

TObject

TPersistent

TCollectionItem

TCustomRVInfo⁽⁶⁹³⁾

TCustomRVFontOrParaInfo⁽⁶⁹³⁾

TCustomRVFontInfo⁽⁶⁹⁶⁾

Properties

The most of properties are inherited from TCustomRVFontInfo⁽⁶⁹⁶⁾.

The new properties are:

- Jump⁽⁷¹⁴⁾ – "is this a hyperlink?"
- NextStyleNo⁽⁷¹⁴⁾ – index of text style to apply after pressing **Enter** in editor (if style templates are not used⁽²⁵⁶⁾);
- ModifiedProperties⁽⁷¹⁴⁾ may (temporarily) contain a list of properties having values different from values defined in a style templates (of this text style⁽⁶⁹⁶⁾ and paragraph⁽⁷⁰⁵⁾).

Methods

IsEqual⁽⁷¹⁷⁾ compares this text style with another one.

You can use Assign⁽⁷¹⁵⁾ method to assign:

- TFontInfo to TFontInfo;
- TCustomRVFontInfo⁽⁶⁹⁶⁾ to TFontInfo and vice versa;
- TFont to TFontInfo and vice versa.

See Also

- TFontInfos⁽⁶⁷⁵⁾ – class of collection of text styles;
- TParaInfo⁽⁷¹⁸⁾ – class of paragraph style;
- TRVListInfo⁽⁷³¹⁾ – class of list style.

6.2.1.5.3.1 Properties

In TFontInfo

- Jump⁽⁷¹⁴⁾
- ModifiedProperties⁽⁷¹⁴⁾
- NextStyleNo⁽⁷¹⁴⁾

Derived from TCustomRVFontInfo⁽⁶⁹⁶⁾

- BackColor⁽⁶⁹⁸⁾
- BiDiMode⁽⁶⁹⁹⁾
- CharScale⁽⁶⁹⁹⁾
- Charset⁽⁷⁰⁰⁾
- CharSpacing⁽⁷⁰⁰⁾
- Color⁽⁷⁰¹⁾
- EmptyWidth⁽⁷⁰¹⁾
- FontName⁽⁷⁰¹⁾
- HoverBackColor⁽⁷⁰²⁾
- HoverColor⁽⁷⁰²⁾
- HoverEffects⁽⁷⁰³⁾
- HoverUnderlineColor⁽⁷⁰³⁾
- JumpCursor⁽⁷⁰⁴⁾
- Options⁽⁷⁰⁴⁾
- ParaStyleTemplateId⁽⁷⁰⁵⁾
- Protection⁽⁷⁰⁵⁾
- Size⁽⁷⁰⁷⁾
- SizeDouble⁽⁷⁰⁷⁾
- Style⁽⁷⁰⁸⁾
- StyleEx⁽⁷⁰⁸⁾
- SubSuperScriptType⁽⁷⁰⁸⁾
- UnderlineColor⁽⁷⁰⁹⁾
- UnderlineType⁽⁷⁰⁹⁾
- VShift⁽⁷¹⁰⁾

Derived from TCustomRVFontOrParaInfo⁽⁶⁹³⁾

- StyleTemplateId⁽⁶⁹⁶⁾

Derived from TCustomRVInfo⁽⁶⁹³⁾

- Standard⁽⁶⁹⁵⁾

■ StyleName⁶⁹⁶

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:

- HoverColor⁷⁰²;
- HoverBackColor⁷⁰²;
- HoverUnderlineColor⁷⁰³;
- HoverEffects⁷⁰³;
- JumpCursor⁷⁰⁴.

See also:

- RichView hypertext overview⁹².

Demo projects:

- Demos*\Assorted\Hypertext\

This property may contain a list of properties not defined in style templates⁶⁵² of this style.

property ModifiedProperties: TRVFontInfoProperties¹⁰⁰⁸;

(introduced in version 14)

This property is used only if style templates are used²⁵⁶.

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:

- StyleTemplateId⁶⁹⁶;
- ParaStyleTemplateId⁷⁰⁵.

See also properties of TParaInfo:

- ModifiedProperties⁷²⁹.

See also methods of TRVStyleTemplate:

- UpdateModifiedTextStyleProperties⁷⁴⁰;
- ApplyToTextStyle⁷³⁹ (contains an example).

Defines the text style for following paragraphs.

property NextStyleNo: Integer;

This property may contain either an index in Style²⁵³.TextStyles⁶⁵⁴, or the value -1.

This property is used in RichViewEdit⁴⁶¹, only if UseStyleTemplates²⁵⁶=*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`⁽²⁵⁶⁾ = `True`, this property is ignored. See the the description `NextId`⁽⁷³⁶⁾ property of the paragraph's style template⁽⁷⁰⁵⁾ about defining text properties for new paragraphs.

Default value:

-1

See also

- `TParaInfo.NextParaNo`⁽⁷³⁰⁾.

property Unicode: Boolean;

Obsolete property. Always `True`.

Default value:

`True`

See also:

- Unicode in `RichView`⁽¹³⁰⁾.

6.2.1.5.3.2 Methods

In `TFontInfo`

`Assign`⁽⁷¹⁵⁾
`Create`⁽⁷¹⁶⁾
`Draw`⁽⁷¹⁶⁾
`IsEqual`⁽⁷¹⁷⁾

Derived from `TCustomRVFontInfo`⁽⁶⁹⁶⁾

`AssignTo`⁽⁷¹¹⁾

Copies the contents of **Source** to this text style.

procedure `Assign(Source: TPersistent); override;`

Call `Assign` to copy properties of `TFont`, `TCustomRVInfo`⁽⁶⁹³⁾, `TCustomRVFontInfo`⁽⁶⁹⁶⁾, or `TFontInfo` to this text style.

Example:

```
// Copying TFont to text style:
MyRVStyle.TextStyles(654)[0].Assign(MyEdit.Font);

// Copying text style to text style:
MyRVStyle.TextStyles(654)[0].Assign(MyRVStyle.TextStyles(654)[1]);
// or, the same:
MyRVStyle.TextStyles(654)[0] := MyRVStyle.TextStyles(654)[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⁽⁷¹¹⁾, this methods initializes:

```
NextStyleNo(714) := -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⁽⁶⁹⁵⁾ := *False*.

Draws **s** on **Canvas**.

```
procedure Draw(const s: TRVUnicodeString(1032); Canvas: TCanvas;
  ThisStyleNo: Integer; SpaceAtLeft, Left, Top, Width, Height
  {$IFDEF RVUSEBASELINE}
    ,BaseLine
  {$ELSE}
    {$IFDEF RVHIGHFONTS}
      ,DrawOffsY
    {$ENDIF}
  {$ENDIF}): TRVCoord(998);
  TextWidth: TRVCoord(998); SpaceCount: Integer;
  RVStyle: TRVStyle(630);
  DrawState: TRVTextDrawStates(1030);
  Printing, PreviewCorrection, ForMetafile: Boolean;
  ColorMode: TRVColorMode(997); DarkMode: Boolean;
  DefBiDiMode: TRVBiDiMode(993);
  RefCanvas: TCanvas; GraphicInterface: TRVGraphicInterface;
  LeftNoExpIndex, RightNoExpIndex: Integer;
  PSaD: PRVScreenAndDevice(1024); AControl: TControl = nil);
```

(changed in versions 18, 19, 23)

Normally, this procedure is called from TRVStyle⁽⁶³⁰⁾.OnDrawStyleText⁽⁶⁷¹⁾ 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**⁽⁶⁵⁴⁾.

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⁽⁶³⁰⁾ 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⁽⁶²⁵⁾, and TRVPrint.PreviewCorrection⁽⁸²⁶⁾=*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⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TRVReportHelper⁽⁸⁰⁴⁾, TRVPrint⁽⁷⁵⁹⁾, it must be equal to Canvas;
- when drawing in TScaleRichViewEdit (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⁽¹⁰²⁷⁾ values and device pixels. It may be *nil* when applying to the screen. This parameter is necessary to apply CharSpacing⁽⁷⁰⁰⁾ 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⁽¹⁰⁰⁸⁾)
(introduced in version 1.3)

Parameter:

Value – object to compare with this text style. The class of this object is TCustomRVFontInfo⁶⁹⁶ or inherited from it. If **Value** is TFontInfo, values of NextStyleNo⁷¹⁴ 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:

- Standard⁶⁹⁵;
- ModifiedProperties⁷¹⁴;
- StyleName⁶⁹⁶ (if RichViewCompareStyleNames¹⁰⁴⁴=False (default)).

See also:

- TCustomRVFontInfo.IsEqual⁷¹¹ (method of the ancestor class);
- TFontInfos.FindSuchStyle⁶⁸¹;
- TParaInfo.IsEqual⁷³¹.

6.2.1.5.4 TCustomRVParaInfo

This is an ancestor class for TParaInfo⁷²⁷ (class of item in collection of paragraph styles⁶⁴⁸).

TCustomRVParaInfo introduces the most of properties of paragraph style.

Unit RVStyle.

Syntax

```
TCustomRVParaInfo = class(TCustomRVFontOrParaInfo693)
```

Hierarchy

TObject

TPersistent

TCollectionItem

*TCustomRVFontOrParaInfo*⁶⁹³

*TCustomRVInfo*⁶⁹³

Properties

- Alignment, LastLineAlignment⁷¹⁹ – alignment of paragraph (left, right, center, justify or distribute);
- FirstIndent⁷²² – indent of the first line of paragraph, summarized with LeftIndent;
- LeftIndent⁷²² – indent from left margin;
- RightIndent⁷²⁵ – indent from right margin;
- SpaceBefore⁷²⁶ – spacing above the paragraph;
- SpaceAfter⁷²⁶ – spacing below the paragraph;
- Border⁷²¹ – border around the paragraph;
- Background⁷²¹ – background of paragraph;
- Tabs⁷²⁶ – collection of tab stops;
- LineSpacing⁷²², LineSpacingType⁷²³ – line spacing;
- BiDiMode⁷²¹ – default text direction;
- Options⁷²³ – protection, wrapping.
- OutlineLevel⁷²⁵ allows defining headings.

Methods

IsEqual⁽⁷²⁷⁾ compares this paragraph style with another one.

You can use Assign⁽⁷²⁶⁾ method to assign TCustomRVParaInfo to TCustomRVParaInfo.

See Also

Classes:

- TParaInfo⁽⁷²⁷⁾ – class inherited from this class, a paragraph style;
- TParaInfos⁽⁶⁸²⁾ – collection of text styles (TParaInfo⁽⁷²⁷⁾).

Overview:

- TRichView Paragraphs⁽⁹⁵⁾

6.2.1.5.4.1 Properties

In TCustomRVParaInfo

- Alignment⁽⁷¹⁹⁾
- Background⁽⁷²¹⁾
- BiDiMode⁽⁷²¹⁾
- Border⁽⁷²¹⁾
- FirstIndent⁽⁷²²⁾
- LastLineAlignment⁽⁷¹⁹⁾
- LineSpacing⁽⁷²²⁾
- LineSpacingType⁽⁷²³⁾
- LeftIndent⁽⁷²²⁾
- Options⁽⁷²³⁾
- OutlineLevel⁽⁷²⁵⁾
- RightIndent⁽⁷²⁵⁾
- SpaceAfter⁽⁷²⁶⁾
- SpaceBefore⁽⁷²⁶⁾
- Tabs⁽⁷²⁶⁾

Derived from TCustomRVFontOrParaInfo⁽⁶⁹³⁾

- StyleTemplateId⁽⁶⁹⁶⁾

Derived from TCustomRVInfo⁽⁶⁹³⁾

- Standard⁽⁶⁹⁵⁾
- StyleName⁽⁶⁹⁶⁾

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
<i>rvaLeft</i>	The paragraph is aligned to the left side.
<i>rvaRight</i>	The paragraph is aligned to the right side.
<i>rvaCenter</i>	The paragraph is centered.
<i>rvaJustify</i>	All lines of the paragraph (except for the last line) are aligned to the both left and right sides by adding space between words (i.e. by increasing widths of space characters). The last line is aligned according to LastLineAlignment property.
<i>rvaDistribute*</i>	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
<i>rllaDefault</i>	The line is aligned to the left side if the paragraph is aligned from left to right ⁽⁷²¹⁾ , or to the right side otherwise.
<i>rllaLeft</i>	The line is aligned to the left side.
<i>rllaRight</i>	The line is aligned to the right side.
<i>rllaCenter</i>	The line is centered.
<i>rllaJustify</i>	The line is aligned according to Alignment property.

Note: Microsoft Word uses the following combinations:

- *rvaJustify* + *rllaDefault*
- *rvaDistribute* + *rllaJustify*

Default value:

- Alignment: *rvaLeft*
- LastLineAlignment: *rllaDefault*

Defines background for paragraphs of this style (transparent by default)

property Background: TRVBackgroundRect⁽⁷⁴¹⁾ ;

(Introduced in version 1.2)

See TRVBackgroundRect⁽⁷⁴¹⁾ for additional information.

Note: RTF and HTML support only equal Background.BorderOffsets⁽⁷⁴²⁾ and Border⁽⁷²¹⁾.BorderOffsets⁽⁷⁴³⁾. RTF and DocX does not support Opacity⁽⁷⁴²⁾.

See also:

- Paragraph Background⁽⁹⁷⁾.

Specifies default text direction for paragraphs of this style.

property BiDiMode: TRVBiDiMode⁽⁹⁹³⁾ ;

(introduced in version 1.6)

Value	Meaning
<i>rvbdUnspecified</i>	Text direction is inherited from higher level, i.e. from TCustomRichView (see BiDiMode ⁽²²⁴⁾)
<i>rvbdLeftToRight</i>	Default text reading order is left-to-right
<i>rvbdRightToLeft</i>	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⁽¹³²⁾.

Defines a border around paragraphs of this style (none by default)

property Border: TRVBorder⁽⁷⁴²⁾ ;

(Introduced in version 1.2)

See TRVBorder⁽⁷⁴²⁾ for additional information.

Note: RTF and HTML support only equal Background⁽⁷²¹⁾.BorderOffsets⁽⁷⁴²⁾ and Border.BorderOffsets⁽⁷⁴³⁾.

See also:

- Paragraph Background⁽⁹⁷⁾.

Indent of the first line for paragraphs of this style.

property FirstIndent: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ 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⁽²³⁸⁾+LeftIndent⁽⁷²²⁾+FirstIndent.

In paragraphs with bullets or numbering, this property is overridden by FirstIndent⁽⁷⁵¹⁾ of list level.

Default value:

0

See also properties:

- LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾;
- SpaceBefore⁽⁷²⁶⁾, SpaceAfter⁽⁷²⁶⁾.

See also properties of TRVListLevel:

- FirstIndent⁽⁷⁵¹⁾.

Left indent for paragraphs of this style.

property LeftIndent: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns this paragraph style.

Indent for paragraph from left border (assuming that indents are converted to pixels) = TCustomRichView.LeftMargin⁽²³⁸⁾+LeftIndent [+ FirstIndent⁽⁷²²⁾ for the first line]

In paragraphs with bullets or numbering, this property is overridden with LeftIndent⁽⁷⁵⁵⁾ of list level (if BiDiMode⁽⁷²¹⁾ <>rvbdRightToLeft).

Default value:

0

See also properties:

- RightIndent⁽⁷²⁵⁾;
- SpaceBefore⁽⁷²⁶⁾, SpaceAfter⁽⁷²⁶⁾.

See also properties of TRVListLevel:

- LeftIndent⁽⁷⁵⁵⁾.

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⁽⁷²³⁾ property.

If LineSpacingType is *rvIsPercent*, 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⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ 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
<i>rvlsPercent</i>	LineSpacing ⁽⁷²²⁾ defines spacing in % of text height. Extra spacing is added/subtracted below lines.
<i>rvlsSpaceBetween</i>	LineSpacing ⁽⁷²²⁾ is a value which will be added between lines, measured in RVStyle.Units ⁽⁶⁵⁵⁾ . This type of line spacing cannot be exported in RTF and HTML. Extra spacing is added below lines.
<i>rvlsLineHeightAtLeast</i>	LineSpacing ⁽⁷²²⁾ defines minimal line height, measured in RVStyle.Units ⁽⁶⁵⁵⁾ . Extra spacing is added above and below lines.
<i>rvlsLineHeightExact</i>	LineSpacing ⁽⁷²²⁾ defines line height, measured in RVStyle.Units ⁽⁶⁵⁵⁾ . 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
<i>rvpaoNoWrap</i>	<p>Disallows automatic word wrapping in paragraphs of this style.</p> <p>If you use no-wrap paragraphs, it's recommended to add <i>rvoClientTextWidth</i> in RichView.Options ⁽²⁴⁰⁾</p>

Editing and Protection Options:

Option	Meaning
<i>rvpaoReadOnly</i>	<p>Disallows editing paragraphs of this style in TRichViewEdit ⁽⁴⁶¹⁾.</p> <p>This option does not protect from adding new paragraph when the user presses Enter at the beginning or at the end of the paragraph.</p> <p>This option does not protect this paragraph from deletion together with the larger selection.</p>
<i>rvpaoStyleProtect</i>	<p>If set, ApplyParaStyle ⁽⁴⁹⁰⁾, ApplyParaStyleTemplate ⁽⁴⁹¹⁾, ApplyStyleTemplate ⁽⁴⁹³⁾ cannot change style of paragraphs of this style.</p> <p>This option does not protect from ApplyParaStyleConversion ⁽⁴⁹¹⁾, but you can take it into account in OnParaStyleConversion ⁽⁵⁸²⁾ event.</p>
<i>rvpaoDoNotWantReturns</i>	<p>If set, Enter key will be blocked if the caret is inside paragraph of this style.</p>

Printing Options

Option	Meaning
<i>rvpaoKeepLinesTogether</i>	<p>Paragraph of this style is printed on one page, if possible.</p>

Option	Meaning
<i>rvpaoKeepWithNext</i>	Paragraph of this style is printed on the same page as the next paragraph, if possible. Additionally, this option works like <i>rvpaoKeepLinesTogether</i>
<i>rvpaoWidowOrphanControl</i>	Prevents page breaks after the first line and before the last line in paragraphs of this style.

Default value:

[]

See also:

- TFontInfo.Protection⁽⁷⁰⁵⁾;
- TCustomRichView.WordWrap⁽²⁵⁸⁾.
- TRVTableRow.KeepTogether⁽⁹¹⁰⁾

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⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns this paragraph style.

Indent for paragraph from the right border (assuming that indents are converted to pixels) = TCustomRichView.RightMargin⁽²⁴⁴⁾ + RightIndent.

In paragraphs with bullets or numbering, this property is overridden with LeftIndent⁽⁷⁵⁵⁾ of list level, if BiDiMode⁽⁷²¹⁾ = *rvbdRightToLeft*.

Default value:

0

See also properties:

- LeftIndent⁽⁷²²⁾.
- SpaceBefore⁽⁷²⁶⁾, SpaceAfter⁽⁷²⁶⁾.

See also properties of TRVListLevel:

- LeftIndent⁽⁷⁵⁵⁾.

Specifies the size of the spacing below the paragraph.

property SpaceAfter: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ 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⁽⁷²⁶⁾;
- LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾.

Specifies the size of the spacing above the paragraph.

property SpaceBefore: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ 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⁽⁷²⁶⁾;
- LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾.

Defines tab stops and their properties for paragraphs of this style.

property Tabs: TRVTabInfos⁽⁷⁴⁵⁾;

(introduced in v1.9)

This is a sorted collection of TRVTabInfo⁽⁷⁴⁶⁾ (sorted by Position⁽⁷⁴⁸⁾ in ascending order).

Default value:

Empty collection.

See also properties of TRVStyle:

- DefTabWidth⁽⁶³⁶⁾.

6.2.1.5.4.2 Methods

In TCustomRVParaInfo

Assign⁽⁷²⁶⁾

Create⁽⁷²⁷⁾

IsEqual⁽⁷²⁷⁾

Copies the contents of **Source** to this paragraph style.

procedure Assign(Source: TPersistent); **override**;

Call Assign to copy properties of TCustomRVInfo⁽⁶⁹³⁾ or TCustomRVParaInfo to this paragraph style.

See also:

- TParaInfo.Assign⁽⁷³⁰⁾ (TParaInfo⁽⁷²⁷⁾ 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(693):
Standard(695) := True;
StyleName(696) := 'Paragraph Style';
// Properties defined in TCustomRVParaInfo:
Alignment(719) := rvaLeft;
LineSpacingType(723) := 100;
LineSpacing(722) := rvlsPercent;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a paragraph style in code, you should assign Standard⁽⁶⁹⁵⁾ := *False*.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties(1017)): 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⁽¹⁰¹⁷⁾.

The following properties are always ignored when comparing styles:

- Standard⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=*False* (default)).

See also:

- TParaInfo.IsEqual⁽⁷³¹⁾ (TParaInfo⁽⁷²⁷⁾ is inherited from this class);
- TCustomRVFontInfo.IsEqual⁽⁷¹¹⁾.

6.2.1.5.5 TParaInfo

This is a paragraph style. It defines attributes of paragraphs⁽⁹⁵⁾ in TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ controls.

This is an item in the collection of TRVStyle⁽⁶³⁰⁾.ParaStyles⁽⁶⁴⁸⁾. The class of the collection itself is TParaInfos⁽⁶⁸²⁾.

Unit RVStyle.

Syntax

```
TParaInfo = class(TCustomRVParaInfo718)
```

Hierarchy

TObject

TPersistent

TCollectionItem

*TCustomRVInfo*⁶⁹³

*TCustomRVFontOrParaInfo*⁶⁹³

*TCustomRVParaInfo*⁷¹⁸

Properties

The most of properties are inherited from *TCustomRVParaInfo*⁷¹⁸.

The new properties are:

- *DefStyleNo*⁷²⁹ – default text style for paragraph;
- *NextParaNo*⁷³⁰ – index of paragraph style to apply after pressing **Enter** in editor (if style templates are not used²⁹⁶);
- *ModifiedProperties*⁷²⁹ may (temporarily) contain a list of properties having values different from values defined in a style template⁶⁹⁶.

Methods

*IsEqual*⁷³¹ compares this paragraph style with another one.

You can use *Assign*⁷³⁰ method to assign:

- *TParaInfo* to *TParaInfo*;
- *TCustomRVParaInfo*⁷¹⁸ to *TParaInfo* and vice versa.

See Also

- *TParaInfos*⁶⁸² – class of collection of paragraph styles;
- *TFontInfo*⁷¹² – class of text style;
- *TRVListInfo*⁷³¹ – class of list style;
- RichView Paragraphs⁹⁵.

6.2.1.5.5.1 Properties

In *TParaInfo*

- *DefStyleNo*⁷²⁹
- *ModifiedProperties*⁷²⁹
- *NextParaNo*⁷³⁰

Derived from *TCustomRVParaInfo*⁷¹⁸

- *Alignment*⁷¹⁹
- *Background*⁷²¹
- *BiDiMode*⁷²¹
- *Border*⁷²¹
- *FirstIndent*⁷²²
- *LastLineAlignment*⁷¹⁹

- LineSpacing⁷²²
- LineSpacingType⁷²³
- LeftIndent⁷²²
- Options⁷²³
- RightIndent⁷²⁵
- SpaceAfter⁷²⁶
- SpaceBefore⁷²⁶
- Tabs⁷²⁶

Derived from TCustomRVFontOrParaInfo⁶⁹³

- StyleTemplateId⁶⁹⁶

Derived from TCustomRVInfo⁶⁹³

- Standard⁶⁹⁵
- StyleName⁶⁹⁶

Specifies the default text style for paragraphs of this style.

```
property DefStyleNo: Integer;
```

(introduced in version 1.8)

Deprecated! Use style templates⁶⁵² to link text and paragraph properties together.

You can use the `rvsDefStyle1059` constant (= `MaxInt`, a large positive integer value) as a `StyleNo` parameter of `AddNL270` 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`).

`GetItemStyle302` method and `CurTextStyleNo474` return a translated text style (they never return `rvsDefStyle`).

Use `GetItem296(i).StyleNo` and `ActualCurTextStyleNo472` to get the text style as is.

Default value:

-1

This property may contain a list of properties not defined in the style template⁶⁵² of this style.

```
property ModifiedProperties: TRVParaInfoProperties1017;
```

(introduced in version 14)

This property is used only if style templates are used²⁵⁶.

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`⁶⁹⁶.

See also properties of TFontInfo:

- `ModifiedProperties`⁷¹⁴.

See also methods of TRVStyleTemplate:

- `UpdateModifiedParaStyleProperties`⁷⁴⁰;
- `ApplyToParaStyle`⁷³⁸ (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`²⁵³.`ParaStyles`⁶⁴⁸, or the value -1.

This property is used in `RichViewEdit`⁴⁶¹, only if `UseStyleTemplates`²⁵⁶ = `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`²⁵⁶ = `True`, this property is ignored, `NextId`⁷³⁶ property of the paragraph's style template⁶⁹⁶ is used instead.

Default value:

-1

See also:

- `TFontInfo.NextStyleNo`⁷¹⁴.

6.2.1.5.5.2 Methods

In TParaInfo

```
Assign730  
Create731  
IsEqual731
```

Copies the contents of **Source** to this paragraph style.

```
procedure Assign(Source: TPersistent); override;
```

Call `Assign` to copy properties of `TCustomRVInfo`⁶⁹³, `TCustomRVParaInfo`⁷¹⁸, or `TParaInfo` to this paragraph style.

Example:

```
// Copying paragraph style to paragraph style:  
MyRVStyle.ParaStyles648[0].Assign(MyRVStyle.ParaStyles648[1]);  
// or, the same:  
MyRVStyle.ParaStyles648[0] := MyRVStyle.ParaStyles648[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⁽⁷²⁷⁾, this methods initializes:

```
DefStyleNo(729) := -1;
```

```
NextParaNo(730) := -1;
```

Note: usually, when you create a paragraph style in code, you should assign Standard⁽⁶⁹⁵⁾ := False.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties(1017)): 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⁽¹⁰¹⁷⁾.

The following properties are always ignored when comparing styles:

- Standard⁽⁶⁹⁵⁾;
- StyleName⁽⁶⁹⁶⁾ (if RichViewCompareStyleNames⁽¹⁰⁴⁴⁾=False (default));
- ModifiedProperties⁽⁷²⁹⁾.

See also:

- TCustomRVParaInfo.IsEqual⁽⁷²⁷⁾ (method of the ancestor class);
- TParaInfos.FindSuchStyle⁽⁶⁸⁵⁾;
- TFontInfo.IsEqual⁽⁷¹⁷⁾.

6.2.1.5.6 TRVListInfo

This is a paragraph list style. It defines attributes of list markers⁽¹⁷⁴⁾ in TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ controls.

This is an item in the collection of TRVStyle⁽⁶³⁰⁾.ListStyles⁽⁶⁴⁶⁾. The class of the collection itself is TRVListInfos⁽⁶⁸⁵⁾.

Unit RVStyle.

Syntax

```
TRVListInfo = class(TCustomRVInfo(693))
```

Hierarchy

TObject

TPersistent

TCollectionItem

TCustomRVInfo⁽⁶⁹³⁾

Properties

- Levels⁽⁷³²⁾ – collection of list levels (TRVListLevel⁽⁷⁵⁰⁾)
- OneLevelPreview⁽⁷³²⁾ suggests appearance of thumbnail.

Other properties are inherited from TCustomRVInfo⁽⁶⁹³⁾.

Methods

- AllNumbered⁽⁷³³⁾ returns *True* if all levels represent numbered lists;
- HasNumbering⁽⁷³³⁾ returns *True* if at least one of list levels represents a numbered list.

See Also

- TRVListInfos⁽⁶⁸⁵⁾ – class of collection of paragraph list styles;
- TFontInfo⁽⁷¹²⁾ – class of text style;
- TParaInfo⁽⁷¹⁸⁾ – class of paragraph style;
- List markers item type⁽¹⁷⁴⁾.

6.2.1.5.6.1 Properties

In TRVListInfo

- Levels⁽⁷³²⁾
- OneLevelPreview⁽⁷³²⁾

Derived from TCustomRVInfo⁽⁶⁹³⁾

- Standard⁽⁶⁹⁵⁾
- StyleName⁽⁶⁹⁶⁾

Levels of paragraph list style, collection of TRVListLevel⁽⁷⁵⁰⁾ objects.

property Levels: TRVListLevelCollection⁽⁷⁴⁹⁾;

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⁽⁷³²⁾.

6.2.1.5.6.2 Methods

In TRVListInfo

AllNumbered⁽⁷³³⁾

HasNumbering⁽⁷³³⁾

Returns *True* if all levels⁽⁷³²⁾ represent numbered lists

```
function AllNumbered: Boolean;
```

See also:

- HasNumbering⁽⁷³³⁾.

Returns *True* if at least one of list levels⁽⁷³²⁾ represents a numbered list.

```
function HasNumbering: Boolean;
```

See also:

- AllNumbered⁽⁷³³⁾.

6.2.1.5.7 TRVStyleTemplate

This class represents a "real" style controlling properties of TFontInfo⁽⁷¹²⁾ and TParaInfo⁽⁷¹²⁾ objects.

This is an item in the collection of TRVStyle⁽⁶³⁰⁾.StyleTemplates⁽⁶⁵²⁾. The class of the collection itself is TRVStyleTemplateCollection⁽⁶⁸⁸⁾.

Unit RVStyle.

Syntax

```
TRVStyleTemplate = class(TCollectionItem)
```

(introduced in version 14)

Hierarchy

TObject

TPersistent

TCollectionItem

Main Properties

- Kind⁽⁷³⁵⁾ – kind of this style templates (defines it applicability to text or paragraphs);
- Name⁽⁷³⁵⁾ – unique name;
- Id⁽⁷³⁴⁾ – unique identifier;
- TextStyle⁽⁷³⁷⁾, ValidTextProperties⁽⁷³⁸⁾ – text properties;
- ParaStyle⁽⁷³⁷⁾, ValidParaProperties⁽⁷³⁸⁾ – paragraph properties;
- ParentId⁽⁷³⁷⁾ – a reference to a parent style template;
- NextId⁽⁷³⁶⁾ – a reference to a style template for following paragraphs;
- QuickAccess⁽⁷³⁷⁾ – recommends to provide a quick access to this style template in applications UI.

Main Methods

- UpdateModifiedTextStyleProperties⁽⁷⁴⁰⁾ fills the set of properties of the specified text style that were changed after applying this style template;
- UpdateModifiedParaStyleProperties⁽⁷⁴⁰⁾ fills the set of properties of the specified paragraph style that were changed after applying this style template;
- ApplyToTextStyle⁽⁷³⁹⁾ applies this style template to the specified text style;
- ApplyToParaStyle⁽⁷³⁸⁾ applies this style template to the specified paragraph style.

Linking

Each style template has two properties having unique values: `Id` ⁽⁷³⁴⁾ and `Name` ⁽⁷³⁵⁾. `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 ⁽⁶⁹⁶⁾ from `TextStyles` ⁽⁶⁵⁴⁾, `ParaStyles` ⁽⁶⁴⁸⁾, and between `StyleTemplates` are made via `Id` property. `Name` is used when merging style templates of two different documents.

Unlike items in `TRVStyle.TextStyles` ⁽⁶⁵⁴⁾ and `TRVStyle.ParaStyles` ⁽⁶⁴⁸⁾, 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 ⁽⁴⁹³⁾).

See also

- [Styles and style templates](#) ⁽⁹⁸⁾.

6.2.1.5.7.1 Properties

In `TRVStyleTemplate`

- ▶ `Id` ⁽⁷³⁴⁾
- `Kind` ⁽⁷³⁵⁾
- `Name` ⁽⁷³⁵⁾
- ▶ `Next` ⁽⁷³⁶⁾
- `NextId` ⁽⁷³⁶⁾
- `ParaStyle` ⁽⁷³⁷⁾
- ▶ `Parent` ⁽⁷³⁷⁾
- `ParentId` ⁽⁷³⁷⁾
- `QuickAccess` ⁽⁷³⁷⁾
- `TextStyle` ⁽⁷³⁷⁾
- `ValidParaProperties` ⁽⁷³⁸⁾
- `ValidTextProperties` ⁽⁷³⁸⁾

An identifier of this style template.

```
property Id: TRVStyleTemplateId(1027);
```

Value of this property is generated automatically using a random number generator. Each item has an identifier that is unique in the collection ⁽⁶⁸⁸⁾.

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` ⁽⁶⁹⁰⁾ with `CopyIds=True`).

This property is used to refer to this style template within a single `TRVStyle` ⁽⁶³⁰⁾ component:

- another style template may refer to this style-template using `ParentId` ⁽⁷³⁷⁾ property;
- another style template may refer to this style-template using `NextId` ⁽⁷³⁶⁾ property;
- text ⁽⁶⁵⁴⁾ and paragraph ⁽⁶⁴⁸⁾ styles of the same `TRVStyle` may refer to this style template using `StyleTemplateId` ⁽⁶⁹⁶⁾ property.

Why do style templates need two unique properties, **Id** and **Name**⁽⁷³⁵⁾? 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⁽⁶⁸⁸⁾:

- FindById, FindItemById⁽⁶⁹¹⁾.

Specifies the kind of this style template.

type

```
TRVStyleTemplateKind = (rvstkParaText, rvstkPara, rvstkText);
```

property Kind: TRVStyleTemplateKind;

Kind	Meaning
<i>rvstkParaText</i>	This style template can be applied both to text and paragraph styles.
<i>rvstkPara</i>	This style template can be applied to paragraphs styles ⁽⁶⁴⁸⁾ (i.e. paragraph styles can be linked ⁽⁶⁹⁶⁾ to this style template)
<i>rvstkText</i>	This style template can be applied to text styles ⁽⁶⁵⁴⁾ (i.e. text styles can be linked ⁽⁶⁹⁶⁾ to this style template)

Assigning a new value to **Name**⁽⁷³⁵⁾ 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**⁽⁷³⁵⁾ 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⁽¹⁸⁰⁾/endnote⁽¹⁷⁸⁾ characters, and to note references⁽¹⁸⁴⁾;
- "endnote text", "footnote text" to apply to text in Document⁽⁹²⁶⁾ of new footnote/endnote.

See also members of TRVStyleTemplateCollection⁽⁶⁸⁸⁾:

- FindByName, FindItemByName⁽⁶⁹¹⁾;
- DefStyleName⁽⁶⁸⁸⁾.

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(1027);
property Next: TRVStyleTemplate; // read-only
```

This property is used only for style templates applied to paragraphs, so it is ignored if Kind⁽⁷³⁵⁾ = *rvstkText*.

To make some style template a next style template, assign its Id⁽⁷³⁴⁾ to **NextId** of this style template.

NextId equals to Id⁽⁷³⁴⁾ 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**⁽⁷³⁴⁾=**NextId**.

When **Enter is pressed, and style templates are used⁽²⁵⁶⁾:**

- 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⁽⁷¹⁴⁾=*True*), the new text is not linked to any style template, its attributes are changed accordingly, its Jump⁽⁷¹⁴⁾=*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⁽⁶⁵⁴⁾ and ParaStyles⁽⁶⁴⁸⁾ 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 (TCustomRVParaInfo718);
```

property ParaStyle: TRVSTParaInfo;

ParaStyle is ignored if Kind⁷³⁵ = *rvstkText*

Only sub-properties listed in ValidParaProperties⁷³⁸ are taken into account. Values of other sub-properties are ignored.

Defines the style template the current style template is based on.

property ParentId: TRVStyleTemplateId¹⁰²⁷;

property Parent: TRVStyleTemplate; // read-only

To make some style template a parent of this style template, assign its Id⁷³⁴ 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**⁷³⁴ = **ParentId**.

If Kind⁷³⁵ = *rvstkText*, only style templates having Kind⁷³⁵ = *rvstkText* can be its parents.

If Kind⁷³⁵ <> *rvstkText*, only style templates having Kind⁷³⁵ <> *rvstkText* can be its parents.

Circular references are not allowed.

The style template inherits (from parents) properties not listed in its ValidTextProperties⁷³⁸ and ValidParaProperties⁷³⁸.

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 (TCustomRVFontInfo696);
```

property TextStyle: TRVSTFontInfo;

Only sub-properties listed in `ValidTextProperties`⁽⁷³⁸⁾ are taken into account. Values of other sub-properties are ignored.

A set of properties of `ParaStyle`⁽⁷³⁷⁾ defined in this style template.

property `ValidParaProperties: TRVParaInfoProperties`⁽¹⁰¹⁷⁾ ;

Only properties included in this set are used. Values of other `ParaStyle`⁽⁷³⁷⁾'s properties are ignored.

This property is used only if `Kind`⁽⁷³⁵⁾ `<>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`⁽⁷³⁷⁾.

A set of properties of `TextStyle`⁽⁷³⁷⁾ defined in this style template.

property `ValidTextProperties: TRVFontInfoProperties`⁽¹⁰⁰⁸⁾ ;

Only properties included in this set are used. Values of other `TextStyle`⁽⁷³⁷⁾'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`⁽⁷³⁷⁾.

6.2.1.5.7.2 Methods

In `TRVStyleTemplate`

`ApplyToParaStyle`⁽⁷³⁸⁾

`ApplyToTextStyle`⁽⁷³⁹⁾

`UpdateModifiedParaStyleProperties`⁽⁷⁴⁰⁾

`UpdateModifiedTextStyleProperties`⁽⁷⁴⁰⁾

Changes properties of **AParaStyle** according to the properties of this style template.

procedure `ApplyToParaStyle(AParaStyle: TCustomRVParaInfo`⁽⁷¹⁸⁾);

The method changes only properties of **AParaStyle** not listed in **AParaStyle.ModifiedProperties**⁽⁷²⁹⁾. If you want to change all properties, assign **AParaStyle.ModifiedProperties**⁽⁷²⁹⁾ = [] before calling this method. If you want to change only properties controlled by style templates, call `UpdateModifiedParaStyleProperties`⁽⁷⁴⁰⁾ before calling this method.

Applying a style template means applying properties of its `ParaStyle`⁽⁷³⁷⁾ listed in `ValidParaProperties`⁽⁷³⁸⁾, and properties inherited from `Parent`⁽⁷³⁷⁾.

`Kind`⁽⁷³⁵⁾ 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**⁽⁶⁹⁶⁾ = **Id**⁽⁷³⁴⁾. This method may be called for *nil*-objects; in this case, it just assigns **AParaStyle.StyleTemplateId**⁽⁶⁹⁶⁾ = -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(733);
begin
  RVStyle.ListStyles(646).Clear;
  RVStyle.TextStyles(654).Clear;
  RVStyle.TextStyles.Add;
  RVStyle.ParaStyles(648).Clear;
  RVStyle.ParaStyles.Add;
  StyleTemplate := RVStyle.StyleTemplates(652).NormalStyleTemplate(689);
  if StyleTemplate<>nil then begin
    RVStyle.ParaStyles[0].ModifiedProperties(729) := [];
    StyleTemplate.ApplyToParaStyle(RVStyle.ParaStyles[0]);
    RVStyle.TextStyles[0].ModifiedProperties(714) := [];
    TRVStyleTemplate(nil).ApplyToTextStyle(739)(
      RVStyle.TextStyles[0], StyleTemplate);
    RVStyle.TextStyles[0].ParaStyleTemplateId(705) := StyleTemplate.Id(734);
  end;
end;
```

See also:

- **ApplyToTextStyle**⁽⁷³⁹⁾

Changes properties of **ATextStyle** according to the properties of this style template.

```
procedure ApplyToTextStyle(ATextStyle: TCustomRVFontInfo(696);
  ParaStyleTemplate: TRVStyleTemplate);
```

The method changes only properties of **ATextStyle** not listed in **ATextStyle.ModifiedProperties**⁽⁷¹⁴⁾. If you want to change all properties, assign **ATextStyle.ModifiedProperties**⁽⁷¹⁴⁾ = [] before calling this method. If you want to change only properties controlled by style templates, call **UpdateModifiedTextStyleProperties**⁽⁷⁴⁰⁾ before calling this method.

Applying a style template means applying properties of its **TextStyle**⁽⁷³⁷⁾ listed in **ValidTextProperties**⁽⁷³⁸⁾, and properties inherited from **Parent**⁽⁷³⁷⁾.

Kind⁽⁷³⁵⁾ 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**⁽⁶⁹⁶⁾ = Id⁽⁷³⁴⁾. This method may be called for *nil*-objects; in this case, it just assigns **ATextStyle.StyleTemplateId**⁽⁶⁹⁶⁾ = -1.
2. Then it applies **ParaStyleTemplate** (if not *nil*) to remaining properties. The method does NOT assign **ATextStyle.ParaStyleTemplateId**⁽⁷⁰⁵⁾.
3. Then it resets remaining properties of **ATextStyle** to default values.

Example: inserting a hyperlink in rve:TRichViewEdit⁽⁴⁶¹⁾. If possible, this hyperlink is formatted using 'Hyperlink' style template.

```

var
  OldStyleTemplate, NewStyleTemplate, ParaStyleTemplate: TRVStyleTemplate(733);
  TextStyle: TFontInfo(712);
begin
  // making a copy of the current text attributes
  // (of the current text style)
  TextStyle := TFontInfo(712).Create(nil);
  TextStyle.Assign(rve.Style(253).TextStyles(654)[rve.CurTextStyleNo(474)]);
  // searching for 'Hyperlink' style template
  NewStyleTemplate := rve.Style.StyleTemplates(652).FindItemByName(691) (
    'Hyperlink');
  if NewStyleTemplate<>nil then begin
    // 'Hyperlink' style template exists. applying it to TextStyle
    OldStyleTemplate := rve.Style.StyleTemplates.FindItemById(691) (
      TextStyle.StyleTemplateId(734));
    ParaStyleTemplate := rve.Style.StyleTemplates.FindItemById(
      TextStyle.ParaStyleTemplateId(705));
    OldStyleTemplate.UpdateModifiedTextStyleProperties(740) (
      TextStyle, ParaStyleTemplate);
    NewStyleTemplate.ApplyToTextStyle(FontInfo, ParaStyleTemplate);
  end
  else begin
    // 'Hyperlink' style template does not exist.
    // Applying default hyperlink properties
    TextStyle.Color(701) := clBlue;
    TextStyle.Style(708) := [fsUnderline];
  end;
  TextStyle.Jump(714) := True;
  // inserting
  rve.CurTextStyleNo := rve.Style.FindTextStyle(661)(TextStyle);
  rve.InsertStringTag(530)('TRichView.com', 'https://www.trichview.com');
  // cleaning up
  TextStyle.Free;
end;

```

See also:

- **ApplyToParaStyle**⁽⁷³⁸⁾ (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(712);

```

```
ParaStyleTemplate: TRVStyleTemplate);
```

```
procedure UpdateModifiedParaStyleProperties(AParaStyle: TParaInfo727);
```

UpdateModifiedTextStyleProperties compares values of properties of **ATextStyle** and properties of this style template (i.e. properties of **TextStyle**⁷³⁷ listed in **ValidTextProperties**⁷³⁸ and properties inherited from **Parent**⁷³⁷). 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**⁷¹⁴.

UpdateModifiedParaStyleProperties compares values of properties of **AParaStyle** and properties of this style template (i.e. properties of **ParaStyle**⁷³⁷ listed in **ValidParaProperties**⁷³⁸ and properties inherited from **Parent**⁷³⁷). Properties having different values are added in **AParaStyle.ModifiedProperties**⁷²⁹.

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**⁷³⁹ and **ApplyToParaStyle**⁷³⁸ methods.

6.2.1.6 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**⁷¹⁸:

- **TRVBackgroundRect**⁷⁴¹ – a type of **Background**⁷²¹ property, properties of paragraph background;
- **TRVBorder**⁷⁴² – a type of **Border**⁷²¹ property, properties of paragraph border;
- **TRVTabInfos**⁷⁴⁵ – a type of **Tabs**⁷²⁶ property, a collection of **TRVTabInfo**⁷⁴⁶ items, defining properties of tab stops.

6.2.1.6.1 TRVBackgroundRect

This class defines background of paragraph (colored rectangle). This is a class of **Background**⁷²¹ property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBackgroundRect = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- **BorderOffsets**⁷⁴²;
- **Color**⁷⁴²;
- **Opacity**⁷⁴².

See also:

- **TCustomRVParaInfo.Background**⁷²¹;
- **Paragraph Borders and Background**⁹⁷.
- **TRVBoxBackgroundRect**⁹³⁸ (background of a floating box)

6.2.1.6.1.1 Properties

In TRVBackgroundRect

- BorderOffsets⁽⁷⁴²⁾
- Color⁽⁷⁴²⁾
- Opacity⁽⁷⁴²⁾

Distance between the paragraph content and the border of the colored area.

property BorderOffsets: TRVRect⁽⁹⁷⁵⁾ ;

Sides of this rectangle are measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns the paragraph style.

Larger positive values increase padding. Negative values force border to be inside the paragraph content.

Warning:

- Background⁽⁷²¹⁾.BorderOffsets.Top must not exceed SpaceBefore⁽⁷²⁶⁾ property of the same paragraph.
- Background.BorderOffsets.Bottom must not exceed SpaceAfter⁽⁷²⁶⁾ property of the same paragraph.
- Background.BorderOffsets.Left should not exceed LeftIndent⁽⁷²²⁾ property of the same paragraph.
- Background.BorderOffsets.Right should not exceed RightIndent⁽⁷²⁵⁾ property of the same paragraph.

Default value:

0,0,0,0 (border is drawn around the paragraph without gaps)

See also:

- Paragraph Background⁽⁹⁷⁾

Color of paragraph background.

property Color: TRVColor⁽⁹⁹⁶⁾

Default value:

rvclNone⁽¹⁰³⁸⁾ (transparent background)

Opacity of paragraph background

property Opacity: TRVOpacity⁽¹⁰¹⁵⁾

(introduced in version 16)

This property is used if Color⁽⁷⁴²⁾ *<>c/None*.

Default value:

100000 (i.e. 100%)

6.2.1.6.2 TRVBorder

This class defines border around paragraph. This is a class of Border⁽⁷²¹⁾ property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBorder = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- BorderOffsets⁽⁷⁴³⁾ – distance between paragraph content and border;
- Color⁽⁷⁴⁴⁾ – border color;
- InternalWidth⁽⁷⁴⁴⁾ – distance between border lines (for double and triple borders);
- Style⁽⁷⁴⁴⁾ – border style;
- VisibleBorders⁽⁷⁴⁴⁾ hides/shows border sides;
- Width⁽⁷⁴⁴⁾ – line width.

See also:

- TCustomRVParaInfo.Border⁽⁷²¹⁾;
- Paragraph Borders and Background⁽⁹⁷⁾.

6.2.1.6.2.1 Properties

In TRVBorder

- BorderOffsets⁽⁷⁴³⁾
- Color⁽⁷⁴⁴⁾
- InternalWidth⁽⁷⁴⁴⁾
- Style⁽⁷⁴⁴⁾
- VisibleBorders⁽⁷⁴⁴⁾
- Width⁽⁷⁴⁴⁾

Spacing between paragraph content and border.

```
property BorderOffsets: TRVRect(975);
```

Sides of this rectangle are measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns the paragraph style.

Larger positive values increase spacing. Negative values force the border to be inside the paragraph content.

Warning:

- Border⁽⁷²¹⁾.BorderOffsets.Top must not exceed SpaceBefore⁽⁷²⁶⁾ property of the same paragraph.
- Border.BorderOffsets.Bottom must not exceed SpaceAfter⁽⁷²⁶⁾ property of the same paragraph.
- Border.BorderOffsets.Left should not exceed LeftIndent⁽⁷²²⁾ property of the same paragraph.
- Border.BorderOffsets.Right should not exceed RightIndent⁽⁷²⁵⁾ property of the same paragraph.

Default value:

0,0,0,0

See also:

- Paragraph Borders⁽⁹⁷⁾.

Color of border

property Color: TRVColor⁹⁹⁶

This property cannot be used to hide the border. To hide border, set Style⁷⁴⁴ = *rvbsNone*.

Default value:

*rvclWindowText*¹⁰³⁸

Spacing between lines in double and triple borders.

property InternalWidth: TRVStyleLength¹⁰²⁷;

This value is measured in Units⁶⁵⁵ of TRVStyle⁶³⁰ component which owns this paragraph style.

For example, if Style⁷⁴⁴ = *rvbDouble*, InternalWidth=1, and Width⁷⁴⁴ =1, and RVStyle.Units⁶⁵⁵ = *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.Units⁶⁵⁵ = *rvstuPixels*, otherwise it could be too small.

Style of a border

property Style: TRVBorderStyle⁹⁹⁴;

See TRVBorderStyle⁹⁹⁴ for possible values.

Default value:

rvbNone (no border)

Defines which sides of border are visible.

property VisibleBorders: TRVBooleanRect⁹⁶¹;

Default value:

True, True, True, True (all sides are visible, but border is still invisible because its Style⁷⁴⁴ = *rvbNone* by default)

Width of border lines.

property Width: TRVStyleLength¹⁰²⁷;

This value is measured in Units⁶⁵⁵ of TRVStyle⁶³⁰ 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 Style⁷⁴⁴ = *rvbSingle*.

Thick lines (Style=*rvbThickInside* or *rvbThickOutside*) have double width.

Default value:

1

Note: this default value assumes that RVStyle.Units⁶⁵⁵ = *rvstuPixels*, otherwise it could be too small.

6.2.1.6.3 TRVTabInfos

Collection of tab stops (`TRVTabInfo`⁷⁴⁶) and their properties.

This is a type of `TRVStyle`⁶³⁰.`ParaStyles`⁶⁴⁸[`i`].`Tabs`⁷²⁶ 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`⁷⁴⁸ 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`⁷⁴⁸, it is moved to the proper place.

Main properties:

- `Items`⁷⁴⁵;
- `Count`.

6.2.1.6.3.1 Properties

In `TRVTabInfos`

`Items`⁷⁴⁵

Derived from `TCollection`

`Count`

Provides indexed access to the items in the collection of tab stops for the paragraph.

```
property Items[Index: Integer]: TRVTabInfo746;
```

This is the default property of `TRVTabInfos`, so you can write

```
MyRVStyle.ParaStyles648[i].Tabs726[j]
```

instead of

```
MyRVStyle.ParaStyles[i].Tabs.Items[j].
```

6.2.1.6.3.2 Methods

In `TRVTabInfos`

`AddFrom`⁷⁴⁶

`DeleteList`⁷⁴⁶

`Find`⁷⁴⁶

Intersect⁽⁷⁴⁶⁾

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⁽⁷⁴⁸⁾ property)

The collection remains sorted.

Deletes tab stops at the specified **Positions**

```
procedure DeleteList(Positions: TRVIntegerList(974));
```

All tab stops having Position⁽⁷⁴⁸⁾ 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⁽⁷⁴⁸⁾ 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(745));
```

6.2.1.6.4 TRVTabInfo

Defines properties for a tab stop. This is a type of item in TRVStyle⁽⁶³⁰⁾.ParaStyles⁽⁶⁴⁸⁾[i].Tabs⁽⁷²⁶⁾ collection.

Unit RVStyle.

Syntax

```
TRVTabInfo = class(TCollectionItem)
```

(introduced in version 1.9)

Hierarchy

TObject

TPersistent

TCollectionItem

Properties

Position⁽⁷⁴⁸⁾ – distance of this tab stop from LeftMargin⁽²³⁸⁾;

Align⁽⁷⁴⁷⁾ – alignment of tab relating to the followed text;

Leader⁽⁷⁴⁷⁾ – text to fill this tabulator.

6.2.1.6.4.1 Properties

In TRVTabInfo

- [Align](#) ⁽⁷⁴⁷⁾
- [Leader](#) ⁽⁷⁴⁷⁾
- [Position](#) ⁽⁷⁴⁸⁾

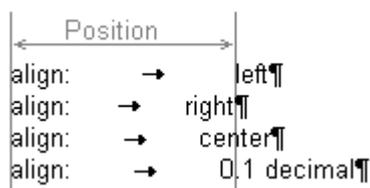
Defines a tab stop alignment (relative to the text after this tab stop)

type

```
TRVTabAlign = (
    rvtaLeft, rvtaRight,
    rvtaCenter, rvtaDecimal);
```

property `Align: TRVTabAlign;`

Image:



Alignments are inverted for right-to-left paragraphs ⁽¹³²⁾.

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](#) ⁽⁷⁴⁸⁾.

Repeated text to fill the tabulator space.

property `Leader: TRVUnicodeString` ⁽¹⁰³²⁾;

(changed in version 18)

Only the following values of this property can be saved in RTF and DocX:

Character	Description
.	dot
-	hyphen

Character	Description
—	underscore character
·	#\$00B7, middle dot
=	equals sign

Default value:

" (empty string)

Defines distance of this tab stop from LeftMargin⁽²³⁸⁾.

property Position: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns the paragraph style with this tab stop.

For right-to-left paragraphs⁽¹³²⁾, the distance is measured from RightMargin⁽²⁴⁴⁾.

Collections of tab stops (TRVStyle⁽⁶³⁰⁾.ParaStyles⁽⁶⁴⁸⁾[i].Tabs⁽⁷²⁶⁾) 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(726)[i].Position := 10;
// index of this tab stop may be changed!
ParaStyle.Tabs(726)[i].Align(747) := rvtaRight;
```

Correct:

```
ParaStyle.Tabs(726)[i].Align(747) := rvtaRight;
ParaStyle.Tabs(726)[i].Position := 10;
```

or

```
with ParaStyles.Tabs(726)[i] do
begin
  Position := 10;
  Align(747) := rvtaRight;
end;
```

Default value:

0

6.2.1.7 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⁽⁷³¹⁾:

- `TRVListLevelCollection`⁽⁷⁴⁹⁾ – a type of `Levels`⁽⁷³²⁾ property, a collection of `TRVListLevel`⁽⁷⁵⁰⁾ items, defining properties of one level of a paragraph list (a bullet or a number)

6.2.1.7.1 TRVListLevelCollection

Collection of list levels. Each level (`TRVListLevel`⁽⁷⁵⁰⁾) defines properties for bullet or numbering of paragraphs.

This is a class of `Levels`⁽⁷³²⁾ property of list styles⁽⁷³¹⁾.

Unit `RVStyle`.

Syntax

```
TRVListLevelCollection = class(TCollection)
```

Hierarchy

TObject

TPersistent

TCollection

Main Properties

`Items`⁽⁷⁴⁹⁾ – array of `TRVListLevel`⁽⁷⁵⁰⁾.

6.2.1.7.1.1 Properties

In TRVListLevelCollection

`Items`⁽⁷⁴⁹⁾

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(750);
```

Each level defines properties for bullet or numbering of paragraphs.

6.2.1.7.1.2 Methods

In TRVListLevelCollection

`Add`⁽⁷⁴⁹⁾

You can also use methods derived from `TCollection`

Adds a new style to the collection of list levels.

```
function Add: TRVListLevel(750);
```

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` array.

`Add` returns the new collection item.

6.2.1.7.2 TRVListLevel

One level of paragraph list, defines properties for bullet or numbering. This is a class of item in the `Levels` collection of list style.

Unit `RVStyle`.

Syntax

```
TRVListLevel = class(TCollectionItem)
```

Hierarchy

TObject

TPersistent

TCollectionItem

Properties

Layout:

- `FirstIndent`, `LeftIndent` override the corresponding properties of paragraph style;
- `MarkerIndent` – indent of marker;
- `MarkerAlignment` – position of marker relative to `MarkerIndent`.

List marker:

- `ListType` – type of marker (bullet, number, image, etc.);
- `Font` – font for text markers;
- `Picture` – picture, used if `ListType` = `rvlstPicture`;
- `ImageList`, `ImageIndex` – used if `ListType` in [`rvlstImageList`, `rvlstImageListCounter`];
- `ImageWidth`, `ImageHeight` – used in FireMonkey if `ListType` in [`rvlstImageList`, `rvlstImageListCounter`];
- `FormatString` – format string for displaying text markers.

Numbering:

- `StartFrom` – initial counter value for numbered lists;
- `Options` – numbering options for this list level.

6.2.1.7.2.1 Properties

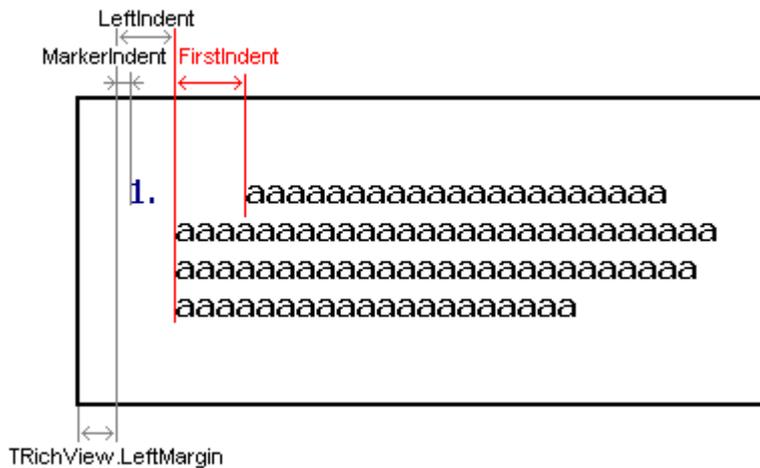
In TRVListLevel

- `FirstIndent`
- `Font`
- `FormatString`
- `ImageHeight`
- `ImageIndex`
- `ImageList`
- `ImageWidth`

- LeftIndent⁽⁷⁵⁵⁾
- ListType⁽⁷⁵⁵⁾
- MarkerAlignment⁽⁷⁵⁷⁾
- MarkerIndent⁽⁷⁵⁷⁾
- Options⁽⁷⁵⁸⁾
- Picture⁽⁷⁵⁹⁾
- StartFrom⁽⁷⁵⁹⁾

Indent for the first line of paragraph text, relative to LeftIndent⁽⁷⁵⁵⁾.

```
property FirstIndent: TRVStyleLength(1027);
```



FirstIndent Example

This value is measured in Units⁽⁶⁵⁵⁾ of TRVStyle⁽⁶³⁰⁾ component which owns the list style.

This value can be negative.

This property overrides FirstIndent⁽⁷²²⁾ 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⁽⁶⁵⁵⁾ = *rvstuPixels*, otherwise it could be too small.

See also properties:

- LeftIndent⁽⁷⁵⁵⁾,
- MarkerIndent⁽⁷⁵⁷⁾.

See also properties of TCustomRichView:

- LeftMargin⁽²³⁸⁾.

Font for text markers.

type

```
TRVMarkerFont = class (TFont);
```

```
property Font: TRVMarkerFont;
```

This font is used for the following list types ⁽⁷⁵⁵⁾:

- *rvlstBullet*,
- *rvlstDecimal*,
- *rvlstDecimalLeadingZero*,
- *rvlstLowerAlpha*,
- *rvlstUpperAlpha*,
- *rvlstLowerRoman*,
- *rvlstUpperRoman*,
- *rvlstLowerGreek*,
- *rvlstUnicodeBullet*.

Default value:

'Arial', 8

Defines format for text list markers.

property FormatString: TRVUnicodeString⁽¹⁰³²⁾;

This property is used for the following list types ⁽⁷⁵⁵⁾:

- *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 - -----"1" ----- 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⁽⁷⁵⁵⁾=*rvlstUnicodeBullet*, this text is displayed.

Since version 18, this property does not make sense, because FormatString⁽⁷⁵²⁾ is Unicode as well.

Default value:

" (empty string)

See also properties:

- FormatString⁽⁷⁵²⁾.

Image index for bullets that show image from the image list

property ImageIndex: Integer;

If ListType⁽⁷⁵⁵⁾=*rvlstImageList*, the marker of this list level is the ImageIndex-th image from ImageList⁽⁷⁵⁴⁾.

If ListType⁽⁷⁵⁵⁾=*rvlstImageListCounter*, the marker of this list level is the (Counter+ImageIndex-1)-th image from ImageList⁽⁷⁵⁴⁾, there Counter is a value of counter for the given list marker.

FireMonkey: the component chooses the image that fits in ImageWidth x ImageHeight⁽⁷⁵⁴⁾.

List counter is numbered from StartFrom⁽⁷⁵⁹⁾.

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⁽⁷⁵⁹⁾.

Image list for image of list marker

```
property ImageList: TCustomImageList;
```

ImageList is used if ListType⁽⁷⁵⁵⁾ is *rvlstImageList* or *rvlstImageListCounter*.

When saving lists of this type in RVF⁽¹²⁴⁾, RichView stores value of ImageList.Tag. When loading, OnRVFImageListNeeded⁽³⁹³⁾ 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⁽⁷⁵⁴⁾.

See also properties:

- ImageIndex⁽⁷⁵³⁾;
- ImageWidth, ImageHeight⁽⁷⁵⁴⁾ [FMX];
- Picture⁽⁷⁵⁹⁾.

Desired size of the list marker image.

```
property ImageWidth: TRVStyleLength(1027);
```

```
property ImageHeight: TRVStyleLength(1027);
```

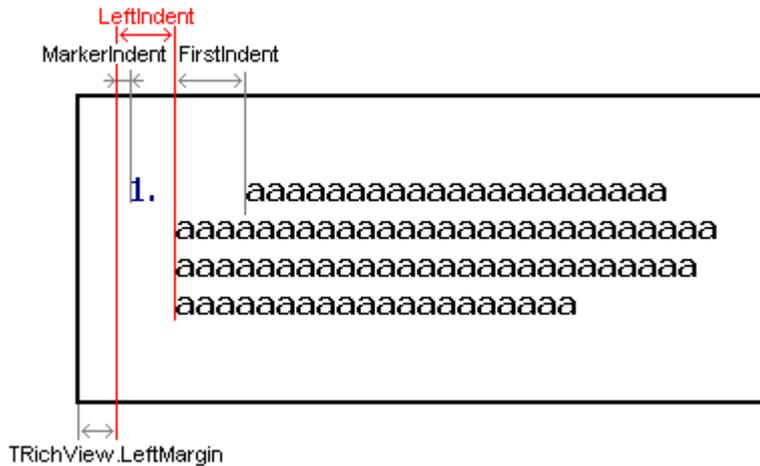
Currently, these properties are used only in FireMonkey, if ListType⁽⁷⁵⁵⁾ = *rvlstImageList* or *rvlstImageListCounter*. The editor chooses the image that fits in the specified size. Zero values are interpreted as 16 device pixels.

See also properties:

- ImageIndex⁽⁷⁵³⁾;
- ImageList⁽⁷⁵⁴⁾.

Left indent of paragraph text.

```
property LeftIndent: TRVStyleLength1027;
```



LeftIndent Example

This value is measured in Units⁶⁵⁵ of TRVStyle⁶³⁰ component which owns the list style.

This value can be negative, but it's not recommended.

This property overrides LeftIndent⁷²² of paragraph style for bulleted/numbered paragraphs. For right-to-left paragraphs¹³², list marker is positioned to the right side of paragraph, and this property overrides RightIndent⁷²⁵ of paragraph style instead.

Default value:

0

See also properties:

- FirstIndent⁷⁵¹
- MarkerIndent⁷⁵⁷.

See also properties of TCustomRichView:

- LeftMargin²³⁸.

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
<i>rvlstBullet</i>	Unnumbered text (from <code>FormatString</code> ⁽⁷⁵²⁾ property). See also <code>Font</code> ⁽⁷⁵¹⁾ .
<i>rvlstPicture</i>	Picture (from <code>Picture</code> ⁽⁷⁵⁹⁾ property)
<i>rvlstImageList</i>	Image from an image list (<code>ImageList</code> ⁽⁷⁵⁴⁾ and <code>ImageIndex</code> ⁽⁷⁵³⁾ properties). FireMonkey: the component chooses the image that fits in <code>ImageWidth</code> x <code>ImageHeight</code> ⁽⁷⁵⁴⁾ .

Numbered List Types

List Type	Meaning
<i>rvlstDecimal</i>	1, 2, 3,... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstDecimalLeadingZero</i>	01, 02, 03, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstLowerAlpha</i>	a, b, c, ..., z, aa, ab, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstUpperAlpha</i>	A, B, C, ..., Z, AA, AB, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstLowerRoman</i>	i, ii, iii, iv, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstUpperRoman</i>	I, II, III, IV, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstLowerGreek</i>	α, β, γ, ..., ω, αα, αβ, ... (see <code>FormatString</code> ⁽⁷⁵²⁾ , <code>Font</code> ⁽⁷⁵¹⁾)
<i>rvlstImageListCounter</i>	Image from an image list, depending on a counter (<code>ImageList</code> ⁽⁷⁵⁴⁾ and <code>ImageIndex</code> ⁽⁷⁵³⁾ properties). FireMonkey: the component chooses the image that fits in <code>ImageWidth</code> x <code>ImageHeight</code> ⁽⁷⁵⁴⁾ .

Obsolete List Types

List Type	Meaning
<i>rvlstUnicodeBullet</i>	Unnumbered Unicode text (from <code>FormatStringW</code> ⁽⁷⁵³⁾ property). Obsolete value. Use <i>rvlstBullet</i> .

Default value:

rvlstBullet

Defines marker alignment relative to the MarkerIndent ⁽⁷⁵⁷⁾

type

```
TRVMarkerAlignment = (rvmaLeft, rvmaRight, rvmaCenter);
```

property MarkerAlignment: TRVMarkerAlignment;



MarkerAlignment Example

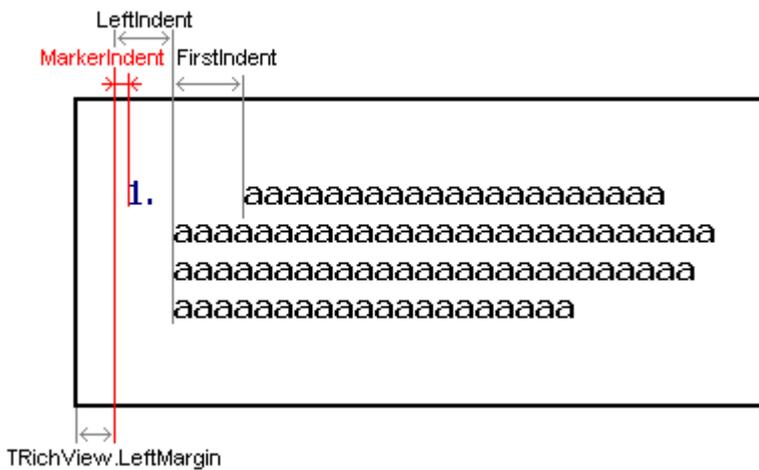
For right-to-left paragraphs ⁽¹³²⁾, list marker position is reversed.

Default value:

rvmaLeft

Indent of the list marker

property MarkerIndent: TRVStyleLength ⁽¹⁰²⁷⁾;



MarkerIndent Example

This value is measured in Units ⁽⁶⁵⁵⁾ of TRVStyle ⁽⁶³⁰⁾ 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 ⁽⁷⁵⁷⁾ property.

Default value:

0

See also properties:

- FirstIndent⁽⁷⁵¹⁾;
- LeftIndent⁽⁷⁵⁵⁾.

See also properties of TCustomRichView:

- LeftMargin⁽²³⁸⁾.

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: TRVPicture975;
```

This property is used if ListType⁷⁵⁵=*rvlstPicture*.

Starting value for marker counter

```
property StartFrom: Integer;
```

For example, if StartFrom for Levels⁷³²[1]=0:

```
1. aaaaa
  0) aaaaa
  1) aaaaa
2. aaaaa
```

Default value:

1

See also:

- ListType⁷⁵⁵.

6.2.1.7.2.2 Methods

In TRVListLevel

HasNumbering⁷⁵⁹

Returns *True* if this list level is numbered

```
function HasNumbering: Boolean;
```

This method returns

Result := ListType⁷⁵⁵ in [*rvlstDecimal*, *rvlstLowerAlpha*, *rvlstUpperAlpha*, *rvlstLowerRoman*, *rvlstUpperRoman*, *rvlstImageListCounter*].

6.2.2 TRVPrint

This component allows printing TRichView²¹⁰, TRichViewEdit⁴⁶¹, TDBRichView⁵⁹⁴, or TDBRichViewEdit⁶⁰⁶. It is Invisible at run-time.

Unit [VCL/FMX] PtblRV / fmxPtblRV.

Syntax

```
TRVPrint = class(TCustomRVPrint822)
```

Hierarchy

TObject

TPersistent

TComponent

TCustomRVPrint⁸²²

How to Use

Printing or drawing pages

1. Assign the source component for printing (`AssignSource`⁽⁷⁷¹⁾ method).
2. (optionally) Assign headers and/or footers (`SetHeader`⁽⁷⁷⁹⁾, `SetFooter`⁽⁷⁷⁸⁾).
3. Format pages (`FormatPages`⁽⁷⁷⁴⁾ method).
4. Now you can:
 - get a number of pages (`PagesCount`⁽⁸²⁶⁾ property);
 - display a print preview (using `TRVPrintPreview`⁽⁶²⁵⁾ component; see also `DrawPreview`⁽⁷⁷³⁾, `MakePreview`⁽⁷⁷⁵⁾, `MakeScaledPreview`⁽⁷⁷⁶⁾, `DrawPage`, `DrawPageAt`⁽⁷⁷³⁾ methods);
 - print the whole document (`Print`⁽⁷⁷⁶⁾ method) or range of pages (`PrintPages`⁽⁷⁷⁷⁾ method).
5. When finished, you can release some temporarily allocated memory (`Clear`⁽⁸²⁷⁾).

PDF saving

1. Check if PDF saving is available (`CanSavePDF`⁽⁸²⁷⁾).
2. Assign the source component for printing (`AssignSource`⁽⁷⁷¹⁾ method).
3. (optionally) Assign headers and/or footers (`SetHeader`⁽⁷⁷⁹⁾, `SetFooter`⁽⁷⁷⁸⁾).
4. Save PDF (`SavePDF`⁽⁷⁷⁸⁾)
5. When finished, you can release some temporarily allocated memory (`Clear`⁽⁸²⁷⁾).

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`⁽⁷⁷⁴⁾.

Alternatively, you can use "virtual printer" to draw pages on any canvas. Assign `VirtualPrinter`⁽⁷⁷⁰⁾. `.Active`⁽⁷⁸⁴⁾ = `True` to use parameters (page size and DPI) specified in `VirtualPrinter`⁽⁷⁷⁰⁾, instead of parameters of the current printer.

Events

You can display some information about the process of paginating and printing:

- `OnFormatting`⁽⁷⁸⁰⁾ occurs during paginating (`FormatPages`⁽⁷⁷⁴⁾ method);
- `OnSendingToPrinter`⁽⁷⁸²⁾ occurs during printing (`Print`⁽⁷⁷⁶⁾ and `PrintPages`⁽⁷⁷⁷⁾ methods).

You can draw additional content using the following events:

- `OnPagePrepaint`⁽⁷⁸²⁾ and `OnPagePostpaint`⁽⁷⁸¹⁾ allow drawing on each page below/over TRichView document;
- `OnPrintComponent`⁽⁸³¹⁾ requests an image of an inserted control for printing.

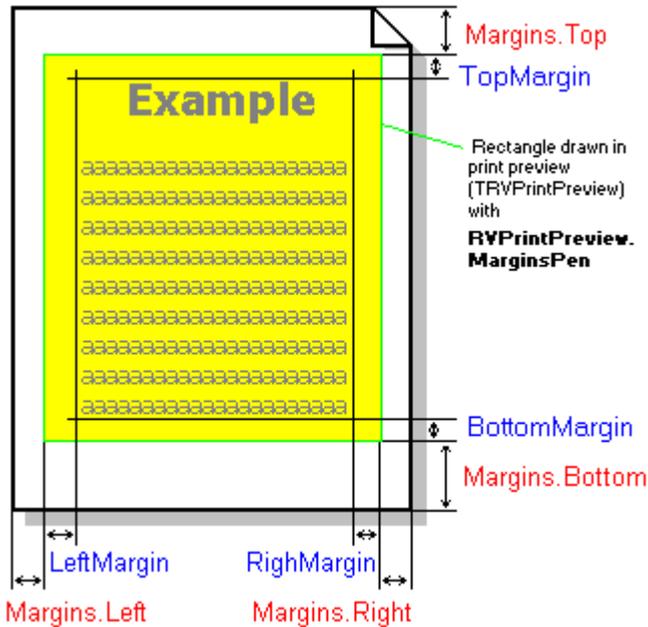
There are also informational events; they occurred when some special item is drawn. You can store coordinates of these items and implement special processing (for example, adding hyperlinks when printing to PDF using some third-party library).

- `OnDrawCheckpoint`⁽⁸³⁰⁾

- OnDrawHyperlink⁽⁸³⁰⁾

Margins

Margins are specified in Margins⁽⁷⁶⁸⁾ property.



Margins in TRVPrint

This picture illustrates the meaning of margin properties. Paragraph indents are not shown for simplification.

Margins⁽⁷⁶⁸⁾ are measured in Units⁽⁷⁷⁰⁾ (shown in red).

TRichView.LeftMargin⁽²³⁸⁾, TRichView.RightMargin⁽²⁴⁴⁾, TRichView.TopMargin⁽²⁵⁵⁾ and TRichView.BottomMargin⁽²²⁵⁾ are measured in “standard” pixels (1 pixel = 1/Style⁽²⁵³⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾ of an inch; shown in blue).

Left and right margins can be swapped for even pages, if MirrorMargins⁽⁷⁶⁹⁾ is set to *True*.

See also ClipMargins⁽⁷⁶⁵⁾ property.

The picture above is for VCL and LCL version of the components.

FireMonkey version uses MarginsStroke⁽⁸³⁷⁾ property instead of MarginsPen property.

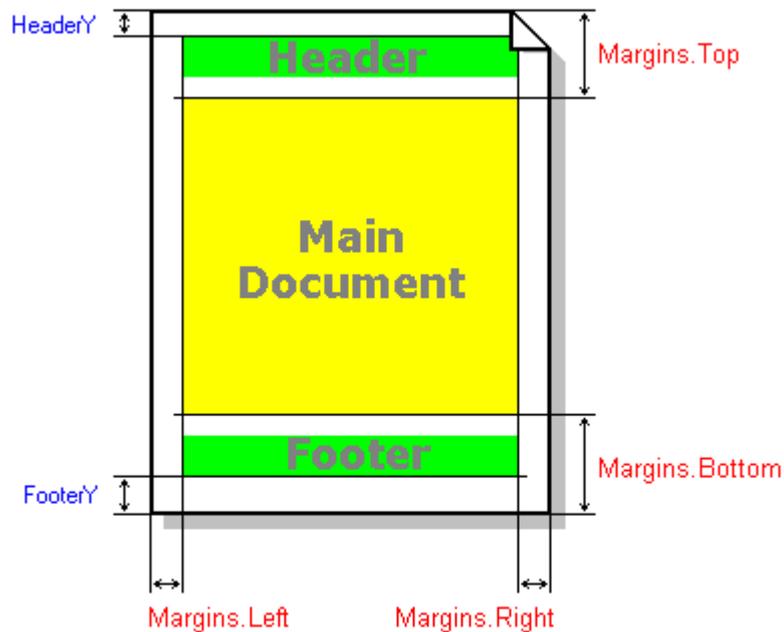
Headers and Footers

TRVPrint supports 3 types of headers and footers:

- normal header and footer
- header and footer for the first page (used only if TitlePage⁽⁷⁶⁹⁾ = *True*)
- header and footer for even pages (used only if FacingPages⁽⁷⁶⁵⁾ = *True*)

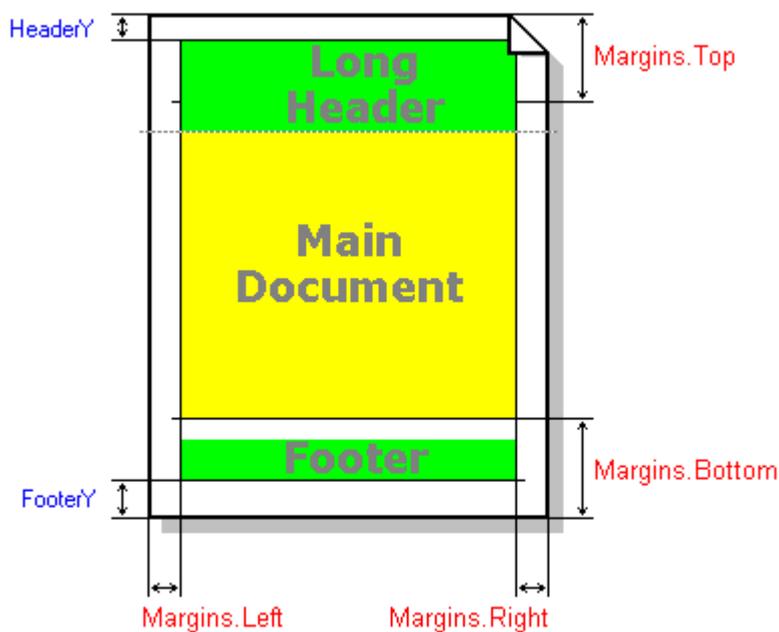
Headers are assigned using SetHeader⁽⁷⁷⁹⁾, footers are assigned using SetFooter⁽⁷⁷⁸⁾ methods.

Any TRichView document can be assigned as a header or footer. Positions of headers and footers are defined by HeaderY⁽⁷⁶⁷⁾ and FooterY⁽⁷⁶⁶⁾.



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` ⁽⁷⁸²⁾ event;
- using `TRVReportHelper` ⁽⁸⁰⁴⁾.

See also: `GetHeaderRect` ⁽⁷⁷⁵⁾, `GetFooterRect` ⁽⁷⁷⁵⁾.

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`⁽⁸²⁴⁾;
- `NoMetafiles`⁽⁸²⁵⁾.

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`⁽²²⁶⁾,
- `BackgroundStyle`⁽²²³⁾
- `BackgroundBitmap`⁽²²²⁾.

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`⁽⁸²⁵⁾ and `MaxPrintedItemNo`⁽⁸²⁴⁾ 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`⁽⁷⁸²⁾ event
3. Header's text boxes⁽¹⁸³⁾ and sidenotes⁽¹⁸¹⁾ having `PositionInText`⁽⁹⁴⁴⁾ = `rvpitBelowText`
4. Header⁽⁷⁷⁹⁾
5. Header's text boxes and sidenotes having `PositionInText` = `rvpitAboveText`
6. Footer's text boxes and sidenotes having `PositionInText` = `rvpitBelowText`
7. Footer⁽⁷⁷⁹⁾
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`⁽⁷⁸¹⁾ 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⁽⁷⁶⁵⁾.

Tips and Tricks

See <https://www.trichview.com/forums/viewtopic.php?p=250>.

See Also

See also:

- UpdatePaletteInfo⁽⁸²⁸⁾ method;
- TRichView.PageBreaksBeforeItems⁽²⁴⁴⁾;
- TRichView.AssignSoftPageBreaks⁽²⁷⁷⁾;
- TRVReportHelper⁽⁸⁰⁴⁾ components;
- RichViewAlternativePicPrint⁽¹⁰⁴³⁾ typed constant;
- RichViewPixelsPerInch⁽¹⁰⁴⁷⁾ typed constant.

Demo projects:

- Demos*\Assorted\Printing\

6.2.2.1 Properties

In TRVPrint

- BestDPI⁽⁷⁶⁵⁾ [FMX]
- ClipMargins⁽⁷⁶⁵⁾
- FixMarginsMode⁽⁷⁶⁶⁾
- FooterY⁽⁷⁶⁶⁾
- HeaderY⁽⁷⁶⁷⁾
- Margins⁽⁷⁶⁸⁾
- MirrorMargins⁽⁷⁶⁹⁾
- ▶ Preview100PercentHeight⁽⁷⁶⁹⁾
- ▶ Preview100PercentWidth⁽⁷⁶⁹⁾
- TitlePage⁽⁷⁶⁹⁾
- Units⁽⁷⁷⁰⁾
- VirtualPrinter⁽⁷⁷⁰⁾

Derived from TCustomRVPrint⁽⁸²²⁾

- ColorMode⁽⁸²²⁾
- DarkMode⁽⁸²³⁾
- ▶ EndAt⁽⁸²³⁾
- ▶ FirstPageNo⁽⁸²³⁾
- IgnorePageBreaks⁽⁸²³⁾
- ▶ LastPageNo⁽⁸²⁴⁾
- MaxPrintedItemNo⁽⁸²⁴⁾
- MetafileCompatibility⁽⁸²⁴⁾
- MetafilePixelsPerInch⁽⁸²⁵⁾

- `MinPrintedItemNo`⁸²⁵
- `NoMetafiles`⁸²⁵
- `PageNoFromNumber`⁸²⁶
- ▶ `PagesCount`⁸²⁶
- `PreviewCorrection`⁸²⁶
- `TransparentBackground`⁸²⁷

6.2.2.1.1 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

6.2.2.1.2 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

6.2.2.1.3 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`⁽⁷⁶⁹⁾;
- `MirrorMargins`⁽⁷⁶⁹⁾.

See also methods:

- `SetHeader`⁽⁷⁷⁹⁾;
- `SetFooter`⁽⁷⁷⁸⁾;
- `AssignDocParameters`⁽⁷⁷¹⁾.

6.2.2.1.4 TRVPrint.FixMarginsMode

Specifies how to work with too small Margins⁽⁷⁶⁸⁾ (less than the printer supports).

type

```
TRVFixMarginsMode = (rvfmmAutoCorrect, rvfmmIgnore);
```

property `FixMarginsMode`: TRVFixMarginsMode;

(introduced in version 10)

Mode	Meaning
<i>rvfmmAutoCorrect</i>	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)
<i>rvfmmIgnore</i>	The specified values of margins are used (document may be cropped)

Default value:

rvfmmAutoCorrect

6.2.2.1.5 TRVPrint.FooterY

Vertical position of a page footer.

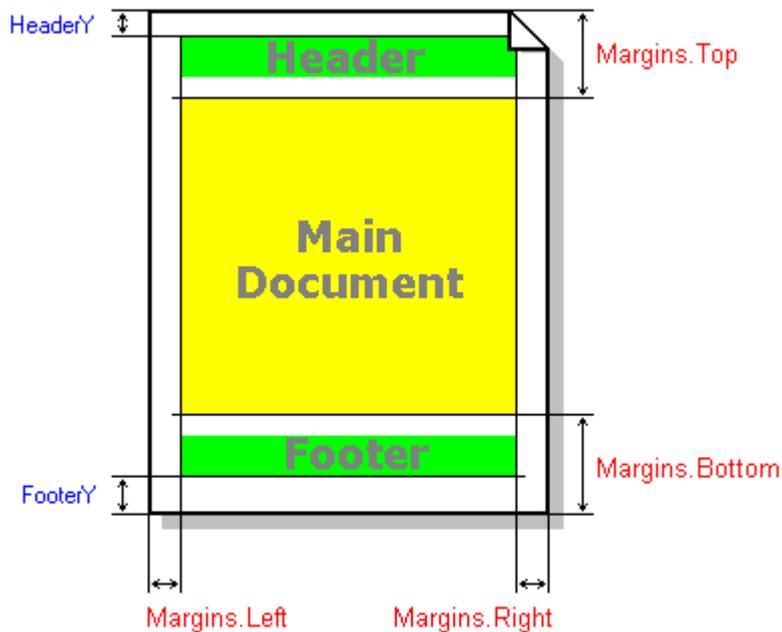
property `FooterY`: TRVLength⁽¹⁰¹⁵⁾;

(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⁽⁷⁷⁰⁾.

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](#)⁽⁷⁶⁷⁾.

See also methods:

- [SetFooter](#)⁽⁷⁷⁸⁾;
- [AssignDocParameters](#)⁽⁷⁷¹⁾.

6.2.2.1.6 TRVPrint.HeaderY

Vertical position of a page header.

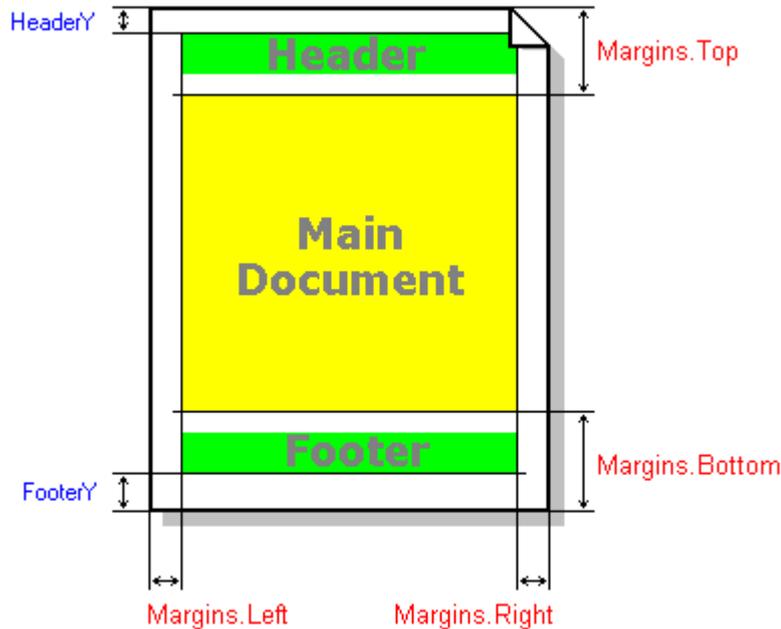
property `HeaderY`: `TRVLength`⁽¹⁰¹⁵⁾;

(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`⁽⁷⁷⁰⁾.

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⁽⁷⁶⁶⁾.

See also methods:

- SetHeader⁽⁷⁷⁹⁾;
- AssignDocParameters⁽⁷⁷¹⁾.

6.2.2.1.7 TRVPrint.Margins

Page margins.

property Margins: TRVUnitsRect⁽⁹⁸⁰⁾;

(introduced in version 15)

Margins are measured in Units⁽⁷⁷⁰⁾.

If MirrorMargins⁽⁷⁶⁹⁾=True, Margins.Left and Margins.Right are swapped for even pages.

You need to reformat pages after changing values of this property, see FormatPages⁽⁷⁷⁴⁾.

See also the picture in the TRVPrint⁽⁷⁵⁹⁾ main topic.

Default value:

(20.0, 20.0, 20.0, 20.0)

See also properties:

- HeaderY⁽⁷⁶⁷⁾;
- FooterY⁽⁷⁶⁶⁾;

See also methods:

- AssignDocParameters⁽⁷⁷¹⁾.

6.2.2.1.8 TRVPrint.MirrorMargins

If set to *True*, `Margins`⁽⁷⁶⁸⁾.Left and `Margins`⁽⁷⁶⁸⁾.Right are swapped for even pages.

property `MirrorMargins`: Boolean;

(introduced in version 1.6)

Default value:

False

See also properties:

- `FacingPages`⁽⁷⁶⁵⁾.

See also methods:

- `AssignDocParameters`⁽⁷⁷¹⁾.

6.2.2.1.9 TRVPrint.Preview100PercentHeight

Height of a page when drawing a preview on the screen at 100% scale, pixels.

property `Preview100PercentHeight`: TRVCoord⁽⁹⁹⁸⁾;

Mainly for internal use.

6.2.2.1.10 TRVPrint.Preview100PercentWidth

Width of a page when drawing a preview on the screen at 100% scale, pixels.

property `Preview100PercentWidth`: TRVCoord⁽⁹⁹⁸⁾;

Mainly for internal use.

6.2.2.1.11 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`⁽⁷⁶⁵⁾.

See also methods:

- `SetHeader`⁽⁷⁷⁹⁾;
- `SetFooter`⁽⁷⁷⁸⁾;
- `AssignDocParameters`⁽⁷⁷¹⁾.

6.2.2.1.12 TRVPrint.Units

Defines measurement units for sizes in TRVPrint.

property Units: TRVUnits⁽¹⁰³²⁾;

(introduced in version 15)

The following properties are measured in Units:

- HeaderY⁽⁷⁶⁷⁾;
- FooterY⁽⁷⁶⁶⁾;
- Margins⁽⁷⁶⁸⁾.

A direct assignment to this property does not change values of these properties. To convert properties, use ConvertToUnits⁽⁷⁷³⁾ method.

If Units⁽⁷⁷⁰⁾ = *rvuPixels*, one unit = 1/RichView.Style⁽²⁵³⁾.UnitsPerInch⁽⁶⁵⁵⁾ of an inch, where RichView is a control assigned for printing by calling AssignSource⁽⁷⁷¹⁾ method. When measuring on paper, they are recalculated according to the printer DPI.

Default value:

rvuMillimeters

6.2.2.1.13 TRVPrint.VirtualPrinter

Defines properties of a "virtual printer"

property VirtualPrinter: TRVVirtualPrinterProperties⁽⁷⁸³⁾;

(introduced in version 17)

If **VirtualPrinter.Active**⁽⁷⁸⁴⁾ = *True*, TRVPrint prepares pages for drawing on any canvas, using page size⁽⁷⁸⁴⁾ and DPI value⁽⁷⁸⁴⁾ 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⁽⁷⁷³⁾ methods. Additionally, if PDF saving engine is available, you can save document as a PDF file⁽⁷⁷⁸⁾.

"Virtual printer" functionality is similar to the functionality provided by TRVReportHelper⁽⁸⁰⁴⁾ 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⁽⁷⁷⁹⁾ and footers⁽⁷⁷⁸⁾;
- a report helper does not support printing text boxes⁽¹⁸³⁾ and side notes⁽¹⁸¹⁾.

6.2.2.2 Methods

In TRVPrint

AssignDocParameters⁽⁷⁷¹⁾
 AssignSource⁽⁷⁷¹⁾
 ContinuousPrint⁽⁷⁷²⁾
 ConvertToUnits⁽⁷⁷³⁾
 Create⁽⁷⁷³⁾
 DrawPageAt⁽⁷⁷³⁾
 DrawPage⁽⁷⁷³⁾

DrawPreview ⁷⁷³
 FormatPages ⁷⁷⁴
 GetFooterRect ⁷⁷⁵
 GetHeaderRect ⁷⁷⁵
 MakePreview ⁷⁷⁵
 MakeScaledPreview ⁷⁷⁶
 Print ⁷⁷⁶
 PrintPages ⁷⁷⁷
 SavePDF ⁷⁷⁸
 SetFooter ⁷⁷⁸
 SetHeader ⁷⁷⁹

Derived from TCustomRVPrint ⁸²²

CanSavePDF ⁸²⁷
 Clear ⁸²⁷
 GetFirstItemOnPage ⁸²⁷
 GetPageNo ⁸²⁸
 UpdatePaletteInfo ⁸²⁸

6.2.2.2.1 TRVPrint.AssignDocParameters

Assigns properties from **DocParameters**.

```
procedure AssignDocParameters(DocParameters: TRVDocParameters 415);
```

(introduced in version 10)

This procedure assigns the following properties:

- Margins ⁷⁶⁸;
- HeaderY ⁷⁶⁷, FooterY ⁷⁶⁶;
- MirrorMargins ⁷⁶⁹;
- FacingPages ⁷⁶⁵;
- TitlePage ⁷⁶⁹.

Sizes are converted from **DocParameters.Units** ⁴²⁰ to Units ⁷⁷⁰ (the value of Units ⁷⁷⁰ property itself is not changed). When converting from/to pixels, this method uses DPI specified in RichView.Style ²⁵³.UnitsPerInch ⁶⁵⁵, where RichView is a control assigned for printing by calling AssignSource ⁷⁷¹ method (or 96, if AssignSource was not called yet).

This method does not assign printer properties (page size and orientation).

6.2.2.2.2 TRVPrint.AssignSource

Assigns document for printing.

```
procedure AssignSource(PrintMe: TCustomRichView 210);
```

You can print document in the following components:

- TRichView ²¹⁰,
- TRichViewEdit ⁴⁶¹,
- TDBRichView ⁵⁹⁴,
- TDBRichViewEdit ⁶⁰⁶.

Document is not copied to RVPrint, so do not modify and do not destroy it until printing is completed.

See also:

- SetHeader⁽⁷⁷⁹⁾;
- SetFooter⁽⁷⁷⁸⁾.

6.2.2.2.3 TRVPrint.ContinuousPrint

Prints the whole document in the current printing job.

procedure ContinuousPrint;

This method is useful if you need to print several documents in one printing job. Otherwise, use Print⁽⁷⁷⁶⁾ or PrintPages⁽⁷⁷⁷⁾ methods.

Do not call this method if VirtualPrinter⁽⁷⁷⁰⁾.Active⁽⁷⁸⁴⁾ = 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(826) := 0;
RVPrint.TransparentBackground(827) := True;
RVPrint.AssignSource(771) (RichView1);
RVPrint.FormatPages(774) (rvdoALL);
RVPrint.ContinuousPrint;
// printing RichView2
RVPrint.StartAt(826) := RVPrint.EndAt(823);
RVPrint.AssignSource(771) (RichView2);
RVPrint.FormatPages(774) (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(771) (RichView1);
RVPrint.FormatPages(774) (rvdoALL);
RVPrint.ContinuousPrint;
// printing RichView2 in landscape orientation
Printer.NewPage;
Printer.Orientation := poLandscape;
RVPrint.AssignSource(771) (RichView2);
RVPrint.FormatPages(774) (rvdoALL);
RVPrint.ContinuousPrint;
Printer.EndDoc;
```

6.2.2.2.4 TRVPrint.ConvertToUnits

Converts properties to new units.

```
procedure ConvertToUnits(AUnits: TRVUnits1032);
```

(introduced in version 15)

The method converts HeaderY⁷⁶⁷, FooterY⁷⁶⁶, Margins⁷⁶⁸ from Units⁷⁷⁰ to **AUnits**, then assigns **AUnits** to Units⁷⁷⁰.

6.2.2.2.5 TRVPrint.Create

A constructor, creates and initializes a TRVPrint instance.

```
constructor Create(AOwner: TComponent); override;
```

Use Create to instantiate TRVPrint object at runtime. TRVPrint objects placed on forms at design time are created automatically. Specify the owner of the new TRVPrint object using the **AOwner** parameter.

6.2.2.2.6 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 PagesCount⁸²⁶.

You need to assign source document for printing (AssignSource⁷⁷¹ method) and format pages (FormatPages⁷⁷⁴ method) before calling this method.

See also methods:

- DrawPreview⁷⁷³.

See also properties:

- PreviewCorrection⁸²⁶;
- PagesCount⁸²⁶.

See also events:

- OnPrintComponent⁸³¹;
- OnPagePrepaint⁷⁸²;
- OnPagePostpaint⁷⁸¹.

6.2.2.2.7 TRVPrint.DrawPreview

Draws print preview of the **pgNo**-th page to **PageRect** rectangle of **Canvas**.

```
procedure DrawPreview(pgNo: Integer; Canvas: TCanvas;  
  const PageRect: TRVCoordRect998);
```

There is more convenient way to display print preview: TRVPrintPreview⁽⁶²⁵⁾ component.

The page image is scaled to fit **PageRect**. To maintain correct proportions, you can use Preview100PercentWidth⁽⁷⁶⁹⁾ and Preview100PercentHeight⁽⁷⁶⁹⁾ properties.

pgNo must be in range from 1 to PagesCount⁽⁸²⁶⁾.

You need to assign source document for printing (AssignSource⁽⁷⁷¹⁾ method) and format pages (FormatPages⁽⁷⁷⁴⁾ method) before calling this method.

To draw unscaled page, use DrawPage and DrawPageAt⁽⁷⁷³⁾ methods.

See also methods:

- MakePreview⁽⁷⁷⁵⁾;
- MakeScaledPreview⁽⁷⁷⁶⁾;
- DrawPage, DrawPageAt⁽⁷⁷³⁾.

See also properties:

- PreviewCorrection⁽⁸²⁶⁾;
- PagesCount⁽⁸²⁶⁾.

See also events:

- OnPrintComponent⁽⁸³¹⁾;
- OnPagePrepaint⁽⁷⁸²⁾;
- OnPagePostpaint⁽⁷⁸¹⁾.

6.2.2.2.8 TRVPrint.FormatPages

Prepares a document for printing (or drawing)

```
function FormatPages(PrintOptions:TRVDisplayOptions(999)): Integer;
```

Please use *rvdoAll* constant for **PrintOptions** parameter:

```
FormatPages(rvdoAll);
```

Document must be assigned to this TRVPrint object using AssignSource⁽⁷⁷¹⁾ method before calling this method.

This method allocates some temporal memory what is used when drawing preview and printing. When you finish, call Clear⁽⁸²⁷⁾ 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⁽⁸²⁷⁾.

See also events:

- OnFormatting⁽⁷⁸⁰⁾.

6.2.2.2.9 TRVPrint.GetFooterRect

Returns coordinates of the footer for the page **PageNo** (from 1), in printer pixels

```
function GetFooterRect (PageNo: Integer): TRVCoordRect998;
```

(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 FormatPages⁷⁷⁴).

This function can be used in OnPagePrepaint⁷⁸² and OnPagePostpaint⁷⁸¹ events.

See also:

- SetFooter⁷⁷⁸;
- GetHeaderRect⁷⁷⁵.

6.2.2.2.10 TRVPrint.GetHeaderRect

Returns coordinates of the header for the page **PageNo** (from 1), in printer pixels

```
function GetHeaderRect (PageNo: Integer): TRVCoordRect998;
```

(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 FormatPages⁷⁷⁴).

This function can be used in OnPagePrepaint⁷⁸² and OnPagePostpaint⁷⁸¹ events.

See also:

- SetHeader⁷⁷⁹;
- GetFooterRect⁷⁷⁵.

6.2.2.2.11 TRVPrint.MakePreview

Draws a preview of the **pgNo**-th page to bitmap **bmp**.

```
procedure MakePreview (pgNo: Integer; bmp: TBitmap);
```

There is more convenient way to display print preview: TRVPrintPreview⁶²⁵ component.

pgNo must be in range from 1 to PagesCount⁸²⁶.

This method changes size of bitmap to (Preview100PercentWidth⁷⁶⁹ x Preview100PercentHeight⁷⁶⁹) and draws print preview of the **pgNo**-th page into it.

You need to assign source document for printing (AssignSource⁷⁷¹ method) and format pages (FormatPages⁷⁷⁴ method) before calling this method.

See also methods:

- MakeScaledPreview⁷⁷⁶;
- DrawPreview⁷⁷³.

See also properties:

- PreviewCorrection⁸²⁶;
- PagesCount⁸²⁶.

See also events:

- OnPrintComponent⁽⁸³¹⁾;
- OnPagePrepaint⁽⁷⁸²⁾;
- OnPagePostpaint⁽⁷⁸¹⁾.

6.2.2.2.12 TRVPrint.MakeScaledPreview

Draws print preview of the **pgNo**-th page to bitmap **bmp**.

```
procedure MakeScaledPreview(pgNo: Integer; bmp: TBitmap);
```

There is more convenient way to display print preview: TRVPrintPreview⁽⁶²⁵⁾ component.

pgNo must be in range from 1 to PagesCount⁽⁸²⁶⁾.

This method does not change size of bitmap, and draws the page in full bitmap size. To maintain correct proportions, you can use Preview100PercentWidth⁽⁷⁶⁹⁾ and Preview100PercentHeight⁽⁷⁶⁹⁾ properties.

You need to assign source document for printing (AssignSource⁽⁷⁷¹⁾ method) and format pages (FormatPages⁽⁷⁷⁴⁾ method) before calling this method.

See also methods:

- MakePreview⁽⁷⁷⁵⁾;
- DrawPreview⁽⁷⁷³⁾.

See also properties:

- PreviewCorrection⁽⁸²⁶⁾;
- PagesCount⁽⁸²⁶⁾.

See also events:

- OnPrintComponent⁽⁸³¹⁾;
- OnPagePrepaint⁽⁷⁸²⁾;
- OnPagePostpaint⁽⁷⁸¹⁾.

6.2.2.2.13 TRVPrint.Print

Prints the whole document

```
procedure Print(Title: String; Copies: Integer; Collate: Boolean;
  PageSet: TRVPageSet(1016) = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo⁽⁸²⁵⁾ and MaxPrintedItemNo⁽⁸²⁴⁾ 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 (AssignSource⁷⁷¹ method) and format pages (FormatPages⁷⁷⁴ method) before calling this method.

```
Print(Title, Copies, Collate, PageSet, Ascending)
```

is equivalent to

```
PrintPages777(1, PagesCount826, Title, Copies, Collate, PageSet, Ascending)
```

Do not call this method if VirtualPrinter⁷⁷⁰.Active⁷⁸⁴ = 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 PagesCount⁸²⁶ is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- PrintPages⁷⁷⁷;

See also events:

- OnSendingToPrinter⁷⁸²;
- OnPrintComponent⁸³¹;
- OnPagePrepaint⁷⁸²;
- OnPagePostpaint⁷⁸¹.

6.2.2.2.14 TRVPrint.PrintPages

Prints the range of pages, from **FirstPgNo** to **LastPgNo**

```
procedure PrintPages(FirstPgNo, LastPgNo: Integer; Title: String;  
Copies: Integer; Collate: Boolean;  
PageSet: TRVPageSet1016 = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo⁸²⁵ and MaxPrintedItemNo⁸²⁴ properties into account.

Parameters

FirstPgNo – the first page to print, must be in range 1..PagesCount⁸²⁶.

LastPgNo – the last page to print, must be in range 1..PagesCount⁸²⁶.

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 (AssignSource⁷⁷¹ method) and format pages (FormatPages⁷⁷⁴ method) before calling this method.

Do not call this method if VirtualPrinter⁷⁷⁰.Active⁷⁸⁴ = 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`⁽⁸²⁶⁾ is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- `Print`⁽⁷⁷⁶⁾.

See also events:

- `OnSendingToPrinter`⁽⁷⁸²⁾;
- `OnPrintComponent`⁽⁸³¹⁾;
- `OnPagePrepaint`⁽⁷⁸²⁾;
- `OnPagePostpaint`⁽⁷⁸¹⁾.

6.2.2.2.15 TRVPrint.SavePDF

Saves a document as a PDF file.

```
function SavePDF(const FileName: TRVUnicodeString(1032);  
PDFInfo: PRVPDFMetadata(1018) = 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⁽¹⁵⁷⁾. You can check availability of a PDF saving engine using `CanSavePDF`⁽⁸²⁷⁾ method.

This method requires an active virtual printer mode⁽⁷⁷⁰⁾. It uses `VirtualPrinter`⁽⁷⁷⁰⁾.`PageWidth` and `PageHeight`⁽⁷⁸⁴⁾ properties.

Internally, this method calls `FormatPages`⁽⁷⁷⁴⁾ method.

Parameters

FileName – output file name

PDFInfo – pointer to `TRVPDFMetadata`⁽¹⁰¹⁸⁾ structure containing PDF creation parameter. This parameter is optional, use `nil` to create PDF with default parameters.

See:

- `TRVReportHelper`⁽⁸⁰⁴⁾.`SavePDF`⁽⁸¹⁰⁾.

6.2.2.2.16 TRVPrint.SetFooter

Assigns a page footer.

```
procedure SetFooter(RVData: TCustomRVData(957);  
FooterType: TRVHFType(1011) = 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`⁽⁷⁷⁴⁾.

Using different footers

A footer for the first page is used only if `TitlePage`⁽⁷⁶⁹⁾ = `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`⁽⁷⁶⁵⁾ = `True`, otherwise it is ignored, and a normal footer is used on both odd and even pages.

Example:

```
var
  rvMain, rvFooter: TRichView(210);
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetFooter(rvFooter.RVData(247));
RVPrint1.TitlePage(769) := True;
// hiding footer on the first page
RVPrint1.SetFooter(nil, rvhftFirstPage);
```

See also:

- `SetHeader`⁽⁷⁷⁹⁾;
- `AssignSource`⁽⁷⁷¹⁾.

See also properties:

- `FooterY`⁽⁷⁶⁶⁾.

See also:

- `TRichView`⁽²¹⁰⁾.`SetFooter`⁽³⁵⁵⁾.

6.2.2.2.17 TRVPrint.SetHeader

Assigns a page header.

```
procedure SetHeader(RVData: TCustomRVData(957);
  FooterType: TRVHFTType(1011)=rvhftNormal);
```

(introduced in version 1.7; modified in version 15)

Parameters:

RVData – document to display as a header of the type specified in **HFTType**; 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.

HFTType – header type (normal/for the first page/for even pages).

This method must be called before `FormatPages`⁽⁷⁷⁴⁾.

Using different headers

A header for the first page is used only if `TitlePage`⁽⁷⁶⁹⁾ = `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`⁽⁷⁶⁵⁾ = `True`, otherwise it is ignored, and a normal header is used on both odd and even pages.

Example:

```

var
  rvMain, rvHeader: TRichView(210);
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetHeader(rvHeader.RVData(247));
RVPrint1.TitlePage(769) := True;
// hiding header on the first page
RVPrint1.SetHeader(nil, rvhftFirstPage);

```

See also:

- SetFooter⁽⁷⁷⁸⁾;
- AssignSource⁽⁷⁷¹⁾.

See also properties:

- HeaderY⁽⁷⁶⁷⁾.

See also:

- TRichView⁽²¹⁰⁾.SetHeader⁽³⁵⁶⁾.

6.2.2.3 Events

In TRVPrint

- OnFormatting⁽⁷⁸⁰⁾
- OnPagePostpaint⁽⁷⁸¹⁾
- OnPagePrepaint⁽⁷⁸²⁾
- OnSendingToPrinter⁽⁷⁸²⁾

Derived from TCustomRVPrint⁽⁸²²⁾

- OnAfterPrintImage⁽⁸²⁹⁾
- OnDrawCheckpoint⁽⁸³⁰⁾
- OnDrawHyperlink⁽⁸³⁰⁾
- OnPrintComponent⁽⁸³¹⁾

6.2.2.3.1 TRVPrint.OnFormatting

Occurs while FormatPages⁽⁷⁷⁴⁾ method is executed.

type

```

TRVPrintingEvent = procedure (Sender: TCustomRichView(210);
  PageCompleted: Integer;
  Step: TRVPrintingStep(1019)) 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 ⁽⁷⁷⁴⁾.

See also events:

- OnSendingToPrinter ⁽⁷⁸²⁾.

6.2.2.3.2 TRVPrint.OnPagePostpaint

Allows to draw on a page (paper and print preview)

type

```
TRVPagePrepaintEvent = procedure (Sender: TRVPrint;  
    PageNo: Integer; Canvas: TCanvas; Preview: Boolean;  
    PageRect, PrintAreaRect: TRVCoordRect (998)) of object;
```

property OnPagePostpaint: TRVPagePrepaintEvent;

(introduced in version 1.7)

Sequence of actions:

1. drawing background (if not transparent);
2. OnPagePrepaint ⁽⁷⁸²⁾;
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 ⁽⁷⁷⁵⁾;
- GetFooterRect ⁽⁷⁷⁵⁾;
- the picture in the TRVPrint ⁽⁷⁵⁹⁾ main topic.

6.2.2.3.3 TRVPrint.OnPagePrepaint

Allows to draw on a page (paper and print preview)

type

```
TRVPagePrepaintEvent = procedure (Sender: TRVPrint;
  PageNo: Integer; Canvas: TCanvas; Preview: Boolean;
  PageRect, PrintAreaRect: TRVCoordRect(998)) 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⁽⁷⁸¹⁾.

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⁽⁷⁷⁵⁾;
- GetFooterRect⁽⁷⁷⁵⁾;
- the picture in the TRVPrint⁽⁷⁵⁹⁾ main topic;
- example how to print pictures in this event: <https://www.trichview.com/forums/viewtopic.php?t=58>.

6.2.2.3.4 TRVPrint.OnSendingToPrinter

Occurs while methods Print⁽⁷⁷⁶⁾ or PrintPages⁽⁷⁷⁷⁾ are executed.

type

```
TRVPrintingEvent = procedure(Sender: TCustomRichView(210);
  PageCompleted: Integer; Step: TRVPrintingStep(1019)) 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⁽⁷⁷⁶⁾;
- PrintPages⁽⁷⁷⁷⁾.

See also events:

- OnFormatting⁽⁷⁸⁰⁾.

6.2.2.4 Classes of properties

Classes of TRVPrint⁽⁷⁵⁹⁾ Properties

- TRVVirtualPrinterProperties⁽⁷⁸³⁾ is a type of TRVPrint.VirtualPrinter⁽⁷⁷⁰⁾ property. It contains properties controlling printing on the specified canvas.

6.2.2.4.1 TRVVirtualPrinterProperties

This is a class of VirtualPrinter⁽⁷⁷⁰⁾ property of TRVPrint⁽⁷⁵⁹⁾.

Unit RVStyle.

Syntax

```
TRVVirtualPrinterProperties = class(TPersistent)
```

Hierarchy

TObject

TPersistent

Main Properties

- Active⁽⁷⁸⁴⁾ enables/disables a virtual printer
- PageWidth, PageHeight⁽⁷⁸⁴⁾
- PixelsPerInch⁽⁷⁸⁴⁾

6.2.2.4.1.1 Properties

In TRVVirtualPrinterProperties

- Active⁽⁷⁸⁴⁾
- PageHeight⁽⁷⁸⁴⁾
- PageWidth⁽⁷⁸⁴⁾
- PixelsPerInch⁽⁷⁸⁴⁾

Enables the virtual printer.

property `Active: Boolean;`

TRVPrint⁽⁷⁵⁹⁾ 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`⁽¹⁰⁴⁷⁾ 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)

6.2.3 TRVSpellChecker

This component checks spelling in TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ 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 TMemor).

Unit [VCL/FMX] RVSpellChecker / fmxRVSpellChecker.

Syntax

```
TRVSpellChecker = class(TComponent)
```

Hierarchy

TObject

TPersistent

TComponent

Availability and Languages

If a spelling checking service is available, `IsAvailable`⁽⁷⁹¹⁾ returns *True*.

`AvailableLanguages`⁽⁷⁸⁶⁾ returns a list of codes of supported languages (dictionaries). To display them to the user, use `FillLanguageNames`⁽⁷⁹⁰⁾ method.

To select a language (dictionary), assign a language code to `Language`⁽⁷⁸⁷⁾ property, or a language index to `LanguageIndex`⁽⁷⁸⁷⁾ property.

The active language (dictionary) is returned by `CurrentLanguage`⁽⁷⁸⁹⁾ or `LanguageIndex`⁽⁷⁸⁷⁾.

Recommended Use

You can check the editor by registering it using `RegisterEditor`⁽⁷⁹²⁾ method.

If you do not use `RichViewActions`, call `Execute`⁽⁷⁸⁹⁾ 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 “sow-level” methods:

- `IsWordValid`⁽⁷⁹¹⁾ – checks a word for spelling errors (can be used in `OnSpellingCheck`⁽⁴⁰⁹⁾ event of editors)
- `Suggest`⁽⁷⁹²⁾ returns suggestions to correct a misspelled word
- `AddToAutoReplaceList`⁽⁷⁸⁸⁾ – “Replace All” command
- `AddToIgnoreList`⁽⁷⁸⁹⁾ – “Ignore All” command
- `AddToDictionary`⁽⁷⁸⁹⁾ – “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`⁽⁷⁹¹⁾.

Auto-correction

You can use `GetAutoCorrection`⁽⁷⁹⁰⁾ 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`⁽⁷⁹¹⁾ 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⁽⁷⁹³⁾ 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⁽⁷⁹³⁾.

See Also

Live spelling checking in TRichView⁽¹³²⁾

6.2.3.1 Properties

In TRVSpellChecker

- ▶ AvailableLanguages⁽⁷⁸⁶⁾
- ▶ DialogShown⁽⁷⁸⁷⁾
- ▶ Language⁽⁷⁸⁷⁾
- ▶ LanguageIndex⁽⁷⁸⁷⁾
- SpellFormStyle⁽⁷⁸⁸⁾

6.2.3.1.1 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 identifies a language (or a dictionary). A format of items depend on the OS.

To switch the active spelling check language, assign Language⁽⁷⁸⁷⁾ or LanguageIndex⁽⁷⁸⁷⁾ property.

To get the active spelling check language, use CurrentLanguage⁽⁷⁸⁹⁾ method or LanguageIndex⁽⁷⁸⁷⁾ property.

To display a list of available languages to the user, use FillLanguageNames⁽⁷⁹⁰⁾ method.

6.2.3.1.2 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⁽⁷⁸⁹⁾.

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)

6.2.3.1.3 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⁽⁷⁸⁶⁾
- 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⁽⁷⁸⁹⁾ property. If you assigned a correct value to **Language**, CurrentLanguage⁽⁷⁸⁹⁾ 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

6.2.3.1.4 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⁽⁷⁸⁶⁾.Count-1.

Assigning to this property changes the active spelling checking language (dictionary). It is identical to assigning Language⁽⁷⁸⁷⁾ = AvailableLanguages⁽⁷⁸⁶⁾[**LanguageIndex**].

This property returns the index of the currently used spelling checking language (dictionary), i.e. index of CurrentLanguage⁽⁷⁸⁹⁾ in AvailableLanguages⁽⁷⁸⁶⁾.

6.2.3.1.5 TRVSpellChecker.SpellFormStyle

Defines an appearance of a spelling checking dialog.

type

```
TRVSpellCheckFormStyle =
  (rvsfsClassic, rvsfsMSWord);
```

property SpellFormStyle: TRVSpellCheckFormStyle;

This dialog is shown by Execute⁽⁷⁸⁹⁾ method.

Value	Meaning
<i>rvsfsClassic</i>	In this dialog, the user can modify a misspelled word.
<i>rvsfsMSWord</i>	In this dialog, the user can modify a misspelled word and and text around it.

6.2.3.2 Methods

In TRVSpellChecker

- AddToAutoreplaceList⁽⁷⁸⁸⁾
- AddToDictionary⁽⁷⁸⁹⁾
- AddToIgnoreList⁽⁷⁸⁹⁾
- CurrentLanguage⁽⁷⁸⁹⁾
- GuessCompletions⁽⁷⁹¹⁾
- Execute⁽⁷⁸⁹⁾
- FillLanguageNames⁽⁷⁹⁰⁾
- GetAutoCorrection⁽⁷⁹⁰⁾
- GetLanguageNameFromCode⁽⁷⁹⁰⁾
- IsAvailable⁽⁷⁹¹⁾
- IsWordValid⁽⁷⁹¹⁾
- RegisterAsService⁽⁷⁹¹⁾ [FMX]
- RegisterEditor⁽⁷⁹²⁾
- StartLiveSpelling⁽⁷⁹²⁾
- Suggest⁽⁷⁹²⁾
- SupportsFeatures⁽⁷⁹³⁾
- UnregisterAsService⁽⁷⁹¹⁾ [FMX]
- UnregisterEditor⁽⁷⁹²⁾

6.2.3.2.1 TRVSpellChecker.AddToAutoreplaceList

Causes occurrences of one word to be replaced by another.

procedure AddToAutoreplaceList(**const** AWordFrom,
AWordTo: TRVUnicodeString⁽¹⁰³²⁾);

This feature is supported not for all platforms. See SupportsFeatures⁽⁷⁹³⁾.

6.2.3.2.2 TRVSpellChecker.AddToDictionary

Add the specified word to the dictionary.

```
function Suggest(const AWord: TRVUnicodeString1032):  
    TRVStringList978;
```

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 SupportsFeatures⁷⁹³.

6.2.3.2.3 TRVSpellChecker.AddToIgnoreList

Ignores the provided word for the rest of this session.

```
procedure AddToIgnoreList(  
    const AWord: TRVUnicodeString1032);
```

Note: a list of ignored words is reset when you switch a spelling checking language.

6.2.3.2.4 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 Language⁷⁸⁷ property. If the assigned language is valid, **CurrentLanguage** will return it.

6.2.3.2.5 TRVSpellChecker.Execute

Starts spelling checking in the specified editor.

```
procedure Execute(Editor: TCustomRichViewEdit461;  
    Scope: TRVEnumScope999 = 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 SpellFormStyle⁷⁸⁸ property.

It's recommended to register this Editor with RegisterEditor⁷⁹² method, because the spell-checker calls LiveSpellingValidateWord³¹⁸ for registered editors when necessary (otherwise, you should process OnSpellFormAction⁷⁹⁴ event and call LiveSpellingValidateWord³¹⁸ 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⁽⁷⁸⁷⁾

6.2.3.2.6 TRVSpellChecker.FillLanguageNames

Add language (dictionary) names to Strings.

```
procedure FillLanguageNames(Strings: TStrings);
```

On return, each string in Strings corresponds to strings of AvailableLanguages⁽⁷⁸⁶⁾, but contain not language codes, but language names.

In other words: for each *i*, Strings.Strings[*i*] = GetLanguageNameFromCode⁽⁷⁹⁰⁾(AvailableLanguages⁽⁷⁸⁶⁾[*i*]).

6.2.3.2.7 TRVSpellChecker.GetAutoCorrection

Returns an auto-correction for the specified word.

```
function GetAutoCorrection(  
  const AWordFrom: TRVUnicodeString(1032);  
  out AWordTo: TRVUnicodeString(1032)): 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⁽⁷⁹³⁾.

6.2.3.2.8 TRVSpellChecker.GetLanguageNameFromCode

Returns a language (dictionary) name for displaying to the user.

```
function GetLanguageNameFromCode(  
  const Language: String): TRVUnicodeString(1032);
```

Input parameter:

Language is a language code, normally one of AvailableLanguages⁽⁷⁸⁶⁾.

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.

6.2.3.2.9 TRVSpellChecker.GuessCompletions

Offers possible completions for **AWordStart**.

```
function GuessCompletions(  
  const AWordStart: TRVUnicodeString1032): TRVStringList978;
```

This feature is supported not for all platforms. See SupportsFeatures⁷⁹³.

6.2.3.2.10 TRVSpellChecker.IsAvailable

Indicates if a spelling checking is available.

```
function IsAvailable: Boolean;
```

Returns *True* if spelling checking can be performed.

Returns *False* otherwise.

6.2.3.2.11 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

6.2.3.2.12 TRVSpellChecker.RegisterAsService, UnregisterAsService

Allows using this spelling checker as spelling checking service for a FireMonkey platform (IFMXSpellCheckerService).

```
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 (Windows and Linux)
- using additional features (such as language selection)

6.2.3.2.13 TRVSpellChecker.RegisterEditor, UnregisterEditor

Registers/unregisters the specified editor in this spelling checker.

```
procedure RegisterEditor(Edit: TCustomRVControl813);
```

```
procedure UnregisterEditor(Edit: TCustomRVControl813);
```

These methods are implemented for the convenience, they are optional.

If you register an editor, the spell-checker:

- assigns the editor's OnSpellingCheck⁴⁰⁹ event to process spelling check in a background thread¹³²;
- calls the editor's ClearLiveSpellingResults²⁷⁹ when changing the active spelling checking language⁷⁸⁷;
- calls the editor's LiveSpellingValidateWord³¹⁸ when the user chooses “Add” or “Ignore All” in the dialog.

Additionally, you can call StartLiveSpelling⁷⁹² to start a background spelling checking in all registered editors.

Input parameter:

Edit – TRichView (TRichView²¹⁰, TRichViewEdit⁴⁶¹, TDBRichView⁵⁹⁴, TDBRichViewEdit⁶⁰⁶) 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.

6.2.3.2.14 TRVSpellChecker.StartLiveSpelling

Starts a background spelling check in all registered editors.

```
procedure StartLiveSpelling;
```

This method calls StartLiveSpelling³⁶⁷ for a registered editors⁷⁹².

6.2.3.2.15 TRVSpellChecker.Suggest

Retrieves spelling suggestions for the specified word.

```
function Suggest(const AWord: TRVUnicodeString1032):  
    TRVStringList978;
```

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.

6.2.3.2.16 TRVSpellChecker.SupportsFeatures

Defines an appearance of a spelling checking dialog.

function SupportsFeatures: TRVSpellFormActions¹⁰¹⁴;

Value	Windows	macOS	iOS	Linux	Android
<i>rvsfaIgnore</i>	+	+	+	+	
<i>rvsfaIgnoreAll</i>	+	+	+	+	
<i>rvsfaChange</i>	+	+	+	+	
<i>rvsfaChangeAll</i> (auto-replacement and auto-correction)	+				
<i>rvsfaAdd</i>	+	+	+		
<i>rvsfaGuessCompletions</i>		+	+		

Note: do not use this function to check availability of spelling checker; use `IsAvailable`⁷⁹¹.

6.2.3.3 Events

In TRVSpellChecker

- OnSpellForm⁷⁹³
- OnSpellFormAction⁷⁹⁴
- OnSpellFormLocalize⁷⁹⁵

6.2.3.3.1 TRVSpellChecker.OnSpellForm

Occurs when a spelling checking dialog is created.

type

```
TRVSpellFormEvent = procedure (
  Sender: TRVSpellChecker784;
  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 SpellFormStyle⁷⁸⁸ = *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*)

6.2.3.3.2 TRVSpellChecker.OnSpellFormAction

Occurs when the user has made a choice in the spelling checking dialog.

type

```
TRVSpellFormActionEvent = procedure (
  Sender: TRVSpellChecker784;
  const AWord, AReplaceTo: TRVUnicodeString1032;
  AAction: TRVSpellFormAction1014) of object;
```

property OnSpellFormAction: TRVSpellFormActionEvent;

This event occurs when Execute⁷⁸⁹ is called, a dialog window is shown, and the user made a choice in this dialog.

AAction	Meaning
<i>rvsfalgnore</i>	The used decided to ignore a single occurrence of AWord
<i>rvsfalgnoreAll</i>	The used decided to ignore all occurrences of AWord
<i>rvsfaChange</i>	The used decided to change one occurrence of AWord to AReplaceTo .
<i>rvsfaChangeAll</i>	The used decided to change all occurrences of AWord to AReplaceTo .
<i>rvsfaAdd</i>	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 RegisterEditor⁷⁹² method, you do not need to perform additional actions. Otherwise, if **AAction** in [*rvsfalgnoreAll*, *rvsfaAdd*], you should call LiveSpellingValidateWord³¹⁸ (**AWord**) for all editors.

6.2.3.3 TRVSpellChecker.OnSpellFormLocalize

Occurs when a spelling checking dialog is created.

type

```
TRVSpellFormEvent = procedure (  
    Sender: TRVSpellChecker(784);  
    Form: TfrmRVCustomSpell_) of object;
```

property OnSpellFormLocalize: TRVSpellFormEvent

This event is used by RichViewActions (TRVSpellInterface component) to localize controls in the dialog.

6.2.4 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⁽⁷⁹⁸⁾ and ExportConverters⁽⁷⁹⁸⁾ properties.

If there are no converters installed, converters lists are empty.

GetImportFilter⁽⁸⁰¹⁾ and GetExportFilter⁽⁸⁰¹⁾ 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⁽⁷⁹⁹⁾ to file in another format (ExportRTF⁽⁸⁰⁰⁾), or you can convert file to RTF Stream (ImportRTF⁽⁸⁰¹⁾).
2. You can export content of TRichView component in a file (ExportRV⁽⁸⁰⁰⁾) or import from file (ImportRV⁽⁸⁰²⁾). Such conversion still uses Stream⁽⁷⁹⁹⁾ internally.

While converting, the component generates OnConverting⁽⁸⁰³⁾ event.

If PreviewMode⁽⁷⁹⁹⁾ 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).

6.2.4.1 Properties

In TRVOfficeConverter

- ▶ ErrorCode⁽⁷⁹⁶⁾
- ExcludeDocXExportConverter⁽⁷⁹⁷⁾
- ExcludeDocXImportConverter⁽⁷⁹⁷⁾
- ExcludeHTMLExportConverter⁽⁷⁹⁷⁾
- ExcludeHTMLImportConverter⁽⁷⁹⁷⁾
- ▶ ExportConverters⁽⁷⁹⁸⁾
- ExtensionsInFilter⁽⁷⁹⁸⁾
- ▶ ImportConverters⁽⁷⁹⁸⁾
- PreviewMode⁽⁷⁹⁹⁾
- ▶ Stream⁽⁷⁹⁹⁾

6.2.4.1.1 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
<i>rvceCnvLoadError</i>	1	error loading converter's DLL
<i>rvceFuncError</i>	2	required function is not found in the converter's DLL
<i>rvceInitError</i>	3	converter initialization failure

Some (not all) error codes returned by converters:

Constant	Value	Meaning
<code>rvceOpenInFileErr</code>	-1	could not open input file
<code>rvceReadErr</code>	-2	error during read
<code>rvceOpenConvErr</code>	-3	error opening conversion file
<code>rvceWriteErr</code>	-4	error during write
<code>rvceInvalidFile</code>	-5	invalid data in conversion file
<code>rvceOpenExceptErr</code>	-6	error opening exception file
<code>rvceWriteExceptErr</code>	-7	error writing exception file
<code>rvceNoMemory</code>	-8	out of memory
<code>rvceInvalidDoc</code>	-9	invalid document
<code>rvceDiskFull</code>	-10	out of space on output
<code>rvceDocTooLarge</code>	-11	conversion document too large for target
<code>rvceOpenOutFileErr</code>	-12	could not open output file
<code>rvceUserCancel</code>	-13	conversion canceled by user
<code>rvceWrongFileType</code>	-14	wrong file type for this converter

6.2.4.1.2 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`⁽⁷⁹⁸⁾.

If `ExcludeDocXExportConverter=True`, DocX export converter is excluded from the list of `ExportConverters`⁽⁷⁹⁸⁾ (TRichView has `SaveDocX`⁽³³³⁾ method allowing to save DocX better than this converter).

Default value

False

6.2.4.1.3 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⁽⁷⁹⁸⁾ (use TRichView.LoadHTML⁽³²²⁾).

If **ExcludeHTMLExportConverter**=*True*, HTML export converter is excluded from the list of ExportConverters⁽⁷⁹⁸⁾ (use TRichView.SaveHTML⁽³³⁵⁾).

Default value

False

6.2.4.1.4 TRVOfficeConverter.ExportConverters

Collection of installed export converters. Export converters convert RTF to foreign formats.

property ExportConverters: TRVOfficeCnvList⁽⁸⁰³⁾;

This is a list of TRVOfficeConverterInfo⁽⁸⁰⁴⁾.

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⁽⁷⁹⁸⁾ property;
- GetExportFilter⁽⁸⁰¹⁾ method.

6.2.4.1.5 TRVOfficeConverter.ExtensionsInFilter

Allows including file extensions in the file filter (GetImportFilter⁽⁸⁰¹⁾ and GetExportFilter⁽⁸⁰¹⁾)

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

6.2.4.1.6 TRVOfficeConverter.ImportConverters

Collection of installed import converters. Import converters convert foreign formats to RTF.

property ImportConverters: TRVOfficeCnvList⁽⁸⁰³⁾;

This is a list of TRVOfficeConverterInfo⁽⁸⁰⁴⁾.

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⁽⁷⁹⁸⁾ property;
- GetImportFilter⁽⁸⁰¹⁾ method.

6.2.4.1.7 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

6.2.4.1.8 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`⁸⁰⁰ and `ImportRTF`⁸⁰¹ work with this stream directly.

The methods `ExportRV`⁸⁰⁰ and `ImportRV`⁸⁰² use this stream internally.

6.2.4.2 Methods

In TRVOfficeConverter

- Create⁷⁹⁹
- Destroy⁷⁹⁹
- ExportRTF⁸⁰⁰
- ExportRV⁸⁰⁰
- GetExportFilter⁸⁰¹
- GetImportFilter⁸⁰¹
- ImportRTF⁸⁰¹
- ImportRV⁸⁰²
- IsValidImporter⁸⁰³

6.2.4.2.1 TRVOfficeConverter.Create

A constructor, creates a new TRVOfficeConverter component.

```
constructor Create(AOwner: TComponent);
```

Use `Create` to instantiate TRVOfficeConverter object at runtime. RVOOfficeConverters placed on forms at design time are created automatically. Specify the owner of the new RVOOfficeConverter using the **AOwner** parameter.

6.2.4.2.2 TRVOfficeConverter.Destroy

Destroys the RVOOfficeConverter

```
destructor Destroy;
```

6.2.4.2.3 TRVOfficeConverter.ExportRTF

Exports RTF from Stream⁽⁷⁹⁹⁾ to file in foreign format.

```
function ExportRTF(const FileName: TRVUnicodeString(1032);  
    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⁽⁷⁹⁸⁾ list).

When calling this method, Stream⁽⁷⁹⁹⁾ must contain RTF (Rich Text Format) document. The method start conversion from Stream⁽⁷⁹⁹⁾.Position and stopped when reached the end of stream.

You can provide a visual indication for the conversion progress using OnConverting⁽⁸⁰³⁾ 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⁽⁸⁰¹⁾;
- ExportRV⁽⁸⁰⁰⁾.

See also properties:

- ErrorCode⁽⁷⁹⁶⁾.

6.2.4.2.4 TRVOfficeConverter.ExportRV

Exports content of **rv** to file in foreign format.

```
function ExportRV(const FileName: TRVUnicodeString(1032);  
    rv: TCustomRichView(210); 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⁽⁷⁹⁸⁾ list)

This method:

1. clears Stream⁽⁷⁹⁹⁾;
2. saves RTF to Stream⁽⁷⁹⁹⁾ using rv.SaveRTFToStream⁽³⁴⁴⁾;
3. performs converting;
4. clears Stream⁽⁷⁹⁹⁾.

You can provide a visual indication for the conversion progress using OnConverting⁽⁸⁰³⁾ 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`⁽⁸⁰¹⁾;
- `ExportRV`⁽⁸⁰⁰⁾.

See also properties:

- `ErrorCode`⁽⁷⁹⁶⁾.

6.2.4.2.5 TRVOfficeConverter.GetExportFilter

Returns string that can be assigned to the Filter property of TSaveDialog.

```
function GetExportFilter: TRVUnicodeString(1032);
```

(changed in version 18)

If `ExtensionsInFilter`⁽⁷⁹⁸⁾ is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- `GetImportFilter`⁽⁸⁰¹⁾ method;
- `ExportConverters`⁽⁷⁹⁸⁾ property.

6.2.4.2.6 TRVOfficeConverter.GetImportFilter

Returns string that can be assigned to the Filter property of TOpenDialog

```
function GetImportFilter: TRVUnicodeString(1032);
```

(changed in version 18)

If `ExtensionsInFilter`⁽⁷⁹⁸⁾ is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- `GetExportFilter`⁽⁸⁰¹⁾ method;
- `ImportConverters`⁽⁷⁹⁸⁾ property.

6.2.4.2.7 TRVOfficeConverter.ImportRTF

Converts data from the file in foreign format to RTF data in `Stream`⁽⁷⁹⁹⁾

```
function ImportRTF(const FileName: TRVUnicodeString(1032);  
  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`⁽⁷⁹⁸⁾ list)

The method clears `Stream`⁽⁷⁹⁹⁾ and writes RTF (Rich Text Format) document in it.

You can provide a visual indication for the conversion progress using `OnConverting`⁽⁸⁰³⁾ 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⁽⁸⁰⁰⁾;
- ImportRV⁽⁸⁰²⁾.

See also properties:

- ErrorCode⁽⁷⁹⁶⁾.

6.2.4.2.8 TRVOfficeConverter.ImportRV

Imports content of file in foreign format to **rv**.

```
function ImportRV(const FileName: TRVUnicodeString(1032);  
  rv: TCustomRichView(210); 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⁽⁷⁹⁸⁾ list)

This method:

1. clears Stream⁽⁷⁹⁹⁾;
2. imports data from the file to Stream⁽⁷⁹⁹⁾;
3. reads data from Stream⁽⁷⁹⁹⁾ in rv, using rv.LoadRTFFromStream⁽³²⁷⁾,
4. clears Stream⁽⁷⁹⁹⁾.

You can provide a visual indication for the conversion progress using OnConverting⁽⁸⁰³⁾ 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⁽⁸⁰¹⁾;
- ExportRV⁽⁸⁰⁰⁾.

See also properties:

- ErrorCode⁽⁷⁹⁶⁾.

6.2.4.2.9 TRVOfficeConverter.IsValidImporter

Checks if the specified file can be imported with the specified converter.

```
function IsValidImporter(const FileName: TRVUnicodeString1032;  
    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 ImportConverters⁷⁹⁸ 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:

- ErrorCode⁷⁹⁶.

6.2.4.3 Events

In TRVOfficeConverter

- OnConverting⁸⁰³

6.2.4.3.1 TRVOfficeConverter.OnConverting

Occurs while converting (in ImportRTF⁸⁰¹, ExportRTF⁸⁰⁰, ImportRV,⁸⁰² ExportRV⁸⁰⁰ 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:

- OnProgress³⁸⁵.

6.2.4.4 Classes of properties

Classes of TRVOfficeConverter⁷⁹⁵ Properties

- TRVOfficeCnvList⁸⁰³ – a list of TRVOfficeConverterInfo⁸⁰⁴ items representing properties of import or export converters. This is a type of ImportConverters⁷⁹⁸ and ExportConverters⁷⁹⁸ properties.

6.2.4.4.1 TRVOfficeCnvList

This class represents a list of installed import⁷⁹⁸ or export⁷⁹⁸ converters.

Unit RVOfficeCnv;

Syntax

```
TRVOfficeCnvList = class (TList)
```

Hierarchy

TObject

TList

Main Properties

The main properties are **Count** (count of converters) and **Items** (array of TRVOfficeConverterInfo⁸⁰⁴).

6.2.4.4.2 TRVOfficeConverterInfo

This class represents an item in list of installed import⁷⁹⁸ and export⁷⁹⁸ 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').

6.2.5 TRVReportHelper

This component is similar to TRVPrint⁷⁵⁹, 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.VirtualPrinter⁷⁷⁰;
- it contains a document, not a reference to it.

Unit [VCL/FMX] RVReport / fmxRVReport.

Syntax

```
TRVReportHelper = class(TCustomRVPrint822)
```

Hierarchy

TObject

TPersistent

TComponent

*TCustomRVPrint*⁸²²

How to Use

1. Load document in RVReportHelper.RichView⁸⁰⁷.
2. Initialize formatting (Init⁸⁰⁹). Specify the page width and canvas.

3. Format the document page by page (FormatNextPage⁸⁰⁹). Pages may have different heights.
4. Draw pages on a canvas (DrawPage⁸⁰⁸ or DrawPageAt⁸⁰⁸). If you need to know positions of hyperlinks and checkpoints, use OnDrawHyperlink⁸³⁰ and OnDrawCheckpoint⁸³⁰ events.

Example (drawing document in a metafile)

```

procedure TForm1.DrawToMetafile(rvh: TRVReportHelper;
  wmf: TMetafile; Width: Integer);
const VERYLARGEVALUE = $FFFFFFFF;
var Canvas: TMetafileCanvas;
begin
  rvh.IgnorePageBreaks823 := True;
  rvh.Init809(Self.Canvas, Width);
  while rvh.FormatNextPage809(VERYLARGEVALUE) do;
  wmf.Width := Width;
  wmf.Height := rvh.GetLastPageHeight809;
  Canvas := TMetafileCanvas.Create(wmf, 0);
  rvh.DrawPage808(1, Canvas, True, rvh.GetLastPageHeight809);
  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:

- MetafileCompatibility⁸²⁴;
- NoMetafiles⁸²⁵.

FireMonkey notes

FireMonkey version

Headers and footers

TRVReportHelper does not support headers and footers.

If you want to store them, you can use TRVReportHelperWithHeaderFooters⁹⁷⁶ class (or use  ScaleRichView).

Demo projects

- Demos*\Assorted\Graphics\ToImage\
- Demos*\Assorted\Printing\ReportHelper\

6.2.5.1 Properties

In TRVReportHelper

- AllowTransparentDrawing⁸⁰⁶
- ▶ RichView⁸⁰⁷
- TargetPixelsPerInch⁸⁰⁶ [VCL,LCL]
- TargetPixelsPerInchX⁸⁰⁶ [FMX]
- TargetPixelsPerInchY⁸⁰⁶ [FMX]

Derived from TCustomRVPrint⁸²²

- ColorMode⁸²²
- DarkMode⁸²³
- ▶ EndAt⁸²³
- ▶ FirstPageNo⁸²³
- IgnorePageBreaks⁸²³
- ▶ LastPageNo⁸²⁴
- MaxPrintedItemNo⁸²⁴
- MetafileCompatibility⁸²⁴
- MetafilePixelsPerInch⁸²⁵
- MinPrintedItemNo⁸²⁵
- NoMetafiles⁸²⁵
- PageNoFromNumber⁸²⁶
- ▶ PagesCount⁸²⁶
- PreviewCorrection⁸²⁶
- TransparentBackground⁸²⁷

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 DrawPage⁸⁰⁸ /DrawPageAt⁸⁰⁸ is *True*.

AllowTransparentDrawing	"Preview"	Result
<i>True</i>	<i>True</i>	Drawing using semitransparent methods
<i>False</i>	<i>True</i>	Drawing semitransparent areas in a bitmap, then drawing this bitmap
<i>True or False</i>	<i>False</i>	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⁽⁸⁰⁹⁾, DrawPage⁽⁸⁰⁸⁾, DrawPageAt⁽⁸⁰⁸⁾ methods.

All sizes are converted from RichView⁽⁸⁰⁷⁾.Style⁽²⁵³⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾ 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⁽⁸⁰⁹⁾.

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⁽²¹⁰⁾;

This RichView object is invisible. Use it to load document and to set properties for loading. This RichView must be linked⁽²⁵³⁾ with some TRVStyle⁽⁶³⁰⁾ 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⁽⁸⁰⁹⁾ and FormatNextPage⁽⁸⁰⁹⁾ instead.

6.2.5.2 Methods

In TRVReportHelper

DrawPage⁽⁸⁰⁸⁾

DrawPageAt⁽⁸⁰⁸⁾

Finished⁽⁸⁰⁹⁾

FormatNextPage⁽⁸⁰⁹⁾

GetLastPageHeight⁽⁸⁰⁹⁾

Init⁽⁸⁰⁹⁾

SavePDF⁽⁸¹⁰⁾

UpdatePageNumbers⁽⁸¹⁰⁾

Derived from TCustomRVPrint⁽⁸²²⁾

Clear⁽⁸²⁷⁾

GetFirstItemOnPage⁽⁸²⁷⁾
 GetPageNo⁽⁸²⁸⁾
 UpdatePaletteInfo⁽⁸²⁸⁾

6.2.5.2.1 TRVReportHelper.DrawPage

Draws the specified page

```
procedure DrawPage(APageNo: Integer; ACanvas: TCanvas;  
  APreview: Boolean; AHeight: TRVCoord(998));
```

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⁽⁸⁰⁹⁾, 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⁽⁸⁰⁹⁾.

If value of TargetPixelsPerInch⁽⁸⁰⁶⁾ is positive, it is assigned to ACanvas.Font.PixelsPerInch.

See also:

- DrawPageAt⁽⁸⁰⁸⁾.

6.2.5.2.2 TRVReportHelper.DrawPageAt

Draws the specified page at the specified coordinates

```
procedure DrawPageAt(Left, Top: TRVCoord(998); APageNo: Integer;  
  ACanvas: TCanvas; APreview: Boolean; AHeight: TRVCoord(998);  
  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⁽⁸⁰⁹⁾, 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`⁸⁰⁹.

If value of `TargetPixelsPerInch`⁸⁰⁶ is positive, it is assigned to `ACanvas.Font.PixelsPerInch`.

See also:

- `DrawPage`⁸⁰⁸.

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`⁸⁰⁹ instead.

6.2.5.2.4 TRVReportHelper.FormatNextPage

Formats next page(s) of document

```
function FormatNextPage(AMaxHeight: TRVCoord998): Boolean;
```

Call this method after `Init`⁸⁰⁹ several times, while it returns *True*.

Parameter:

AMaxHeight – page height, pixels; page width is specified as a parameter of `Init`⁸⁰⁹.

When the page is formatted, it can be drawn (`DrawPage`⁸⁰⁸, `DrawPageAt`⁸⁰⁸).

Sometimes, this method formats more than one page.

This method is fast. The most time consuming method is `Init`⁸⁰⁹.

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: TRVCoord998;
```

This function returns height of document on the last page, when the document is completely formatted (all calls to `FormatNextPage`⁸⁰⁹ are made).

The returned value is measured in **ACanvas** pixels, where **ACanvas** is the parameter of `Init`⁸⁰⁹.

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: TRVCoord998);
```

Parameters:

ACanvas – canvas object used for formatting.

APageWidth – page width, pixels; all pages have the same width.

VCL and LCL: If value of `TargetPixelsPerInch`⁽⁸⁰⁶⁾ is positive, it is assigned to `ACanvas.Font.PixelsPerInch`.

FMX: If **ACanvas** is a printer canvas, assign the active printer DPI to (`TargetPixelsPerInchX`, `TargetPixelsPerInchY`)⁽⁸⁰⁶⁾ properties before calling `Init`.

See also:

- `FormatNextPage`⁽⁸⁰⁹⁾.

6.2.5.2.7 TRVReportHelper.SavePDF

Saves a document in RichView⁽⁸⁰⁷⁾ 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`⁽¹⁵⁷⁾. You can check availability of a PDF saving engine using `CanSavePDF`⁽⁸²⁷⁾ method.

Internally, this method calls `Init`⁽⁸⁰⁹⁾ and `FormatNextPage`⁽⁸⁰⁹⁾ methods.

Parameters

FileName – output file name

PageWidth, PageHeight – page size

PDFInfo – pointer to `TRVPDFMetadata`⁽¹⁰¹⁸⁾ structure containing PDF creation parameter. This parameter is optional, use `nil` to create PDF with default parameters.

See:

- `TRVPrint`⁽⁷⁵⁹⁾.`SavePDF`⁽⁷⁷⁸⁾.

6.2.5.2.8 TRVReportHelper.UpdatePageNumbers

Recalculates page numbers.

```
procedure UpdatePageNumbers;
```

(introduced in version 15)

This method recalculates values of "page number"⁽¹⁸⁵⁾ and "page count"⁽¹⁸⁶⁾ 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`⁽⁸¹¹⁾ event, and call **UpdatePageNumber** before drawing.

6.2.5.3 Events

In TRVReportHelper

- OnGetPageNumber⁽⁸¹¹⁾

Derived from TCustomRVPrint⁽⁸²²⁾

- OnAfterPrintImage⁽⁸²⁹⁾
- OnDrawCheckpoint⁽⁸³⁰⁾
- OnDrawHyperlink⁽⁸³⁰⁾
- OnPrintComponent⁽⁸³¹⁾

6.2.5.3.1 TRVReportHelper.OnGetPageNumber

Request a page number for displaying in "page number" fields⁽¹⁸⁵⁾.

type

```
TRVGetPageNumberEvent = procedure (Sender: TRVReportHelper(804);
  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⁽⁸²⁶⁾), where the "page number" field is located.

DispPageNo – a number that will be displayed in this field. Initially, it is equal to **PageNo+PageNoFromNumber**⁽⁸²⁶⁾-1. You can assign another value to this parameter.

See also:

- UpdatePageNumbers⁽⁸¹⁰⁾

6.2.6 TRVDataSourceLink

A simple component that links data-aware controls inside TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, 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⁽⁸¹²⁾ to data-aware controls when they are inserted in Editor⁽⁸¹²⁾.

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⁽⁸¹²⁾ can be a data-aware control by itself (TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾ 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.

6.2.6.1 Properties

In TRVDataSourceLink

- DataSource⁽⁸¹²⁾
- Editor⁽⁸¹²⁾

6.2.6.1.1 TRVDataSourceLink.DataSource

A TDataSource component for assigning to data-aware controls in Editor⁽⁸¹²⁾.

```
property DataSource: TDataSource;
```

TDataSource provides an interface between a dataset component and data-aware controls in Editor⁽⁸¹²⁾.

Value of this property is assigned automatically to DataSource properties of all data-aware controls when they are inserted in Editor⁽⁸¹²⁾ (if another DataSource is not already assigned to them).

When you assign a new value to this property, existing data-aware controls in Editor⁽⁸¹²⁾ are not affected; this value will be used only for controls that will be inserted in Editor⁽⁸¹²⁾ later. If you want to change DataSource of existing controls, call Execute⁽⁸¹³⁾.

6.2.6.1.2 TRVDataSourceLink.Editor

A TRichView or ScaleRichView component that contains (or will contain) data-aware controls.

```
property Editor: TCustomRVControl(813);
```

When a new data-aware control is inserted in **Editor**, this component assigns DataSource⁽⁸¹²⁾ to its DataSource property.

If you want to change DataSource of controls that are already in **Editor**, call Execute⁽⁸¹³⁾.

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

- TRichView⁽²¹⁰⁾
- TRichViewEdit⁽⁶⁰⁶⁾
- TDBRichView⁽⁵⁹⁴⁾
- TDBRichViewEdit⁽⁶⁰⁶⁾
- TSRichViewEdit (from ScaleRichView)
- TDBSRichViewEdit (from ScaleRichView)

Only one TRVDataSourceLink⁽⁸¹¹⁾ 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.

6.2.6.2 Methods

In TRVDataSourceLink

Execute⁽⁸¹³⁾

6.2.6.2.1 TRVDataSourceLink.Execute

Assigns DataSource⁽⁸¹²⁾ to all data-aware controls in Editor⁽⁸¹²⁾.

```
procedure Execute(OnlyToUndefined: Boolean);
```

Normally, you do not need to call this method, because DataSource⁽⁸¹²⁾ is assigned to all data-aware controls in Editor⁽⁸¹²⁾ 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⁽⁸¹²⁾.

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⁽⁸¹²⁾.

6.3 Components ancestor classes

TRichView Components⁽²⁰⁹⁾ | Ancestor Classes

TRVScroller⁽⁸¹³⁾ – ancestor of main visual components: TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁶⁰⁶⁾.

TCustomRVPrint⁽⁸²²⁾ – ancestor of TRVPrint⁽⁷⁵⁹⁾ and TRVReportHelper⁽⁸⁰⁴⁾

TCustomRVPrintPreview⁽⁸³²⁾ – of TRVPrintPreview⁽⁶²⁵⁾

6.3.1 TRVScroller

This class is an ancestor of all visual components in TRichView family (TRichView⁽²¹⁰⁾, TRichViewEdit⁽⁴⁶¹⁾, TDBRichView⁽⁵⁹⁴⁾, TDBRichViewEdit⁽⁴⁶¹⁾, TRVPrintPreview⁽⁶²⁵⁾).

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
TCustomScrollBar
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*²¹⁰.

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*⁸¹⁹ – position of vertical scrollbar;
- *VScrollMax*⁸¹⁸ (read-only) – maximum value of *VScrollPos*;
- *NoHScroll*⁸¹⁷ – disallows horizontal scrolling;
- *NoVScroll*⁸¹⁷ – disallows vertical scrolling;
- *HScrollPos*⁸¹⁶ – position of horizontal scrollbar, measured in pixels;
- *HScrollMax*⁸¹⁶ (read-only) – maximum value of *HScrollPos*;
- *InplaceEditor*⁸¹⁷ – returns "inplace" editor of some item in document (used in tables for editing cells).

Additional properties for VCL and LCL versions:

- *WheelStep*⁸¹⁹ defines speed of mouse wheel scrolling.

The most important method is:

- *ScrollTo/ScrollToPosition*⁸²⁰, scrolls content of the component vertically to the specified position (in pixels).

6.3.1.1 Properties

In *TRVScroller*

- *BorderStyle*⁸¹⁶ [VCL,LCL]
- ▶ *HScrollMax*⁸¹⁶
- HScrollPos*⁸¹⁶
- *HScrollVisible*⁸¹⁶ [VCL,LCL]

- ▶ InplaceEditor⁸¹⁷
- ▶ NoHScroll⁸¹⁷
- ▶ NoVScroll⁸¹⁷
- Tracking⁸¹⁸
- UseXPThemes⁸¹⁸ [VCL,LCL]
- ▶ VScrollMax⁸¹⁸
- ▶ VScrollPos⁸¹⁹
- VScrollVisible⁸¹⁹ [VCL,LCL]
- WheelStep⁸¹⁹ [VCL,LCL]

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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
- ShowScrollBar [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
<i>bsNone</i>	No visible border
<i>bsSingle</i>	Border, 3D or themed ⁸¹⁸

6.3.1.1.2 TRVScroller.HScrollMax

Maximal possible value for HScrollPos ⁸¹⁶

```
property HScrollMax: TRVCoord 998;
```

See also properties:

- HScrollPos ⁸¹⁶;
- VScrollMax ⁸¹⁸.

See also:

- Scrolling in RichView ¹⁰⁵

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 998;
```

This value is in range 0..HScrollMax ⁸¹⁶.

See also properties:

- VScrollPos ⁸¹⁹.

See also:

- Scrolling in RichView ¹⁰⁵.

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⁽⁸¹⁹⁾.

See also:

- Scrolling in RichView⁽¹⁰⁵⁾.

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⁽⁸⁴⁶⁾ (see "Cell Editing"⁽¹⁹⁷⁾). In future versions, inplace editors can be used for some other types of items.

Inplace editors are derived from TCustomRichViewEdit⁽⁴⁶¹⁾ 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⁽⁴⁸¹⁾.

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

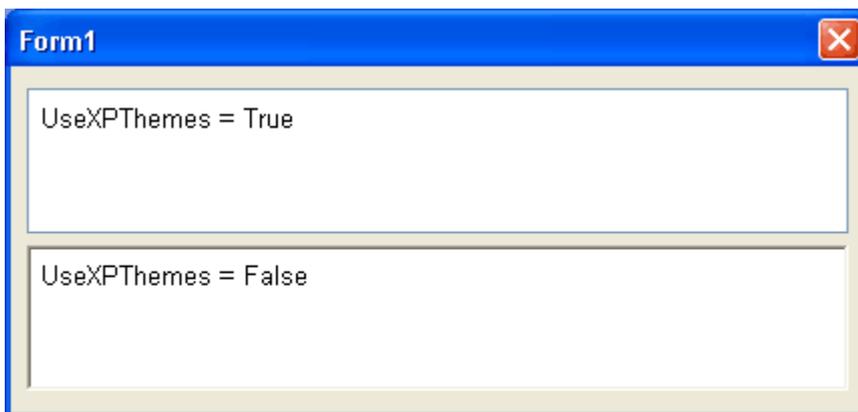
Allows using Windows themes (visual styles)

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:

- BorderStyle⁸¹⁶.

6.3.1.1.9 TRVScroller.VScrollMax

Maximal possible value for VScrollPos⁸¹⁹

property VScrollMax: TRVCoord⁹⁹⁸;

See also properties:

- VScrollPos⁸¹⁹;
- HScrollMax⁸¹⁶.

See also methods:

- ScrollTo⁸²⁰.

See also:

- Scrolling in RichView¹⁰⁵.

6.3.1.1.10 TRVScroller.VScrollPos

Position of vertical scrollbar.

```
property VScrollPos: TRVCoord998;
```

This value is in range 0..VScrollMax⁸¹⁸.

VCL and LCL:

This value is measured in "RVSU"¹⁰⁵, 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:

- HScrollPos⁸¹⁶.

See also methods:

- ScrollTo⁸²⁰.

See also:

- Scrolling in RichView¹⁰⁵.

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 TRichViewEdit⁴⁶¹ 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:

- HScrollVisible⁸¹⁶.

See also:

- Scrolling in RichView¹⁰⁵.

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 $\text{Round}(\text{VSmallStep}^{257} * \text{WheelStep} * \text{suggested} / 3)$ pixels.

Default value:

2

6.3.1.2 Methods

In TRVScroller

Create ⁸²⁰
ScrollTo ⁸²⁰

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 ²¹⁰;
- TRichViewEdit ⁴⁶¹;
- TDBRichView ⁵⁹⁴;
- TDBRichViewEdit ⁴⁶¹;
- TRVPrintPreview ⁶²⁵.

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 ⁹⁹⁸; 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 ⁹⁹⁸; SmoothScroll: Boolean = False);

SmoothScroll is ignored.

See also properties:

- VScrollPos ⁸¹⁹;
- VScrollMax ⁸¹⁸.

See also:

- Scrolling in TRichView ¹⁰⁵.

6.3.1.3 Events

In TRVScroller

- OnHScrolled ⁸²¹

- OnVScrolled⁸²¹

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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:

- OnVScrolled⁸²¹.

See also:

- Scrolling in RichView¹⁰⁵.

6.3.1.3.2 TRVScroller.OnVScrolled

Occurs when document is scrolled vertically

property OnVScrolled: TNotifyEvent;

See also events:

- OnHScrolled⁸²¹.

See also properties of TRichView:

- FirstItemVisible²³⁶;

- LastItemVisible ⁽²³⁸⁾.

See also:

- Scrolling in RichView ⁽¹⁰⁵⁾.

6.3.2 TCustomRVPrint

TCustomRVPrint is an ancestor class of TRVPrint ⁽⁷⁵⁹⁾ and TRVReportHelper ⁽⁸⁰⁴⁾ components.

This is an intermediate class. Do not use it directly in your programs.

Unit [VCL/FMX] PtbIRV / fmxPtbIRV.

Syntax

```
TCustomRVPrint = class(TComponent)
```

Hierarchy

TObject

TPersistent

TComponent

6.3.2.1 Properties

In TCustomRVPrint

- ColorMode ⁽⁸²²⁾
- DarkMode ⁽⁸²³⁾
- ▶ EndAt ⁽⁸²³⁾
- ▶ FirstPageNo ⁽⁸²³⁾
- IgnorePageBreaks ⁽⁸²³⁾
- ▶ LastPageNo ⁽⁸²⁴⁾
- MaxPrintedItemNo ⁽⁸²⁴⁾
- MetafileCompatibility ⁽⁸²⁴⁾
- MetafilePixelsPerInch ⁽⁸²⁵⁾
- MinPrintedItemNo ⁽⁸²⁵⁾
- NoMetafiles ⁽⁸²⁵⁾
- PageNoFromNumber ⁽⁸²⁶⁾
- ▶ PagesCount ⁽⁸²⁶⁾
- PreviewCorrection ⁽⁸²⁶⁾
- StartAt ⁽⁸²⁶⁾
- TransparentBackground ⁽⁸²⁷⁾

6.3.2.1.1 TCustomRVPrint.ColorMode

Sets color mode for printing and drawing preview

```
property ColorMode: TRVColorMode (997) ;
```

(Introduced in version 1.8)

Default values:

- *rvcmColor* for TRVReportHelper ⁽⁸⁰⁴⁾ ;
- *rvcmPrinterColor* for TRVPrint ⁽⁷⁵⁹⁾ .

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⁽⁸⁰⁴⁾, or TRVPrint⁽⁷⁵⁹⁾ with active VirtualPrinter⁽⁷⁷⁰⁾ mode). When printing, black background causes overuse of printer ink.

Default value:

False

See also:

- TCustomRichView⁽²¹⁰⁾.DarkMode⁽²²⁸⁾

6.3.2.1.3 TCustomRVPrint.EndAt

Height of document on the last page, in device pixels.

```
property EndAt: TRVCoord(998);
```

This property does not include margins.

You can use this property only when document is completely formatted (see FormatPages⁽⁷⁷⁴⁾ for TRVPrint, Init⁽⁸⁰⁹⁾ and FormatNextPage⁽⁸⁰⁹⁾ for TRVReportHelper).

For TRVReportHelper, you can use GetLastPageHeight⁽⁸⁰⁹⁾ instead (it's more convenient, because it includes top and bottom margins).

See also properties:

- StartAt⁽⁸²⁶⁾.

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⁽⁸²⁵⁾ and MaxPrintedItemNo⁽⁸²⁴⁾ into account. By default, when these properties define the range including the whole document, FirstPageNo returns 1. If MinPrintedItemNo⁽⁸²⁵⁾>0, this property returns the index of the page containing the MinPrintedItemNo⁽⁸²⁵⁾-th item (see GetPageNo⁽⁸²⁸⁾). If MinPrintedItemNo⁽⁸²⁵⁾>=ItemCount, FirstPageNo returns PagesCount⁽⁸²⁶⁾+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` ⁽²⁴⁴⁾ [];
- `TRVTableItemInfo.Rows[]`.PageBreakBefore ⁽⁹¹⁰⁾.

This property is useful in `TRVReportHelper` ⁽⁸⁰⁴⁾ 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` ⁽⁸²⁵⁾ and `MaxPrintedItemNo` ⁽⁸²⁴⁾ into account. By default, when these properties define the range including the whole document, `LastPageNo` returns `PagesCount` ⁽⁸²⁶⁾. If `MaxPrintedItemNo` ⁽⁸²⁴⁾ is inside the items range (0..Item-1), this property returns the index of the page containing the `MaxPrintedItemNo` ⁽⁸²⁴⁾-th item (see `GetPageNo` ⁽⁸²⁸⁾).

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` ⁽⁸²⁵⁾ to `MaxPrintedItemNo`. By default, this range includes all items.

The value -1 is treated like `Item-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` ⁽⁸²³⁾ to `LastPageNo` ⁽⁸²⁴⁾).

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` ⁽⁶⁵⁴⁾ = *rvtUniscribe*, 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⁽¹⁸⁷⁾) 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⁽¹⁶³⁾ and *hot-pictures*⁽¹⁶⁶⁾) are printed as bitmaps, if NoMetafiles⁽⁸²⁵⁾ = *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⁽⁸²⁴⁾. 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⁽⁸²³⁾ to LastPageNo⁽⁸²⁴⁾).

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⁽¹⁶³⁾ and *hot-pictures*⁽¹⁶⁶⁾) and equations⁽¹⁸⁷⁾ 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` ⁽⁸²⁵⁾

6.3.2.1.12 TCustomRVPrint.PageNoFromNumber

Specifies the number of the first page for "page number" fields ⁽¹⁸⁵⁾.

property `PageNoFromNumber: Integer;`

(introduced in version 15)

Default value:

1

See also:

- `TRVReportHelper.OnGetPageNumber` ⁽⁸¹¹⁾

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` ⁽⁷⁷⁴⁾ for `TRVPrint`, `Init` ⁽⁸⁰⁹⁾ and `FormatNextPage` ⁽⁸⁰⁹⁾ 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` ⁽⁹⁹⁸⁾;

This property is useful (in conjunction with `EndAt` ⁽⁸²³⁾) in `TRVPrint.ContinuousPrint` ⁽⁷⁷²⁾ 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`⁽⁸⁶³⁾ of tables.

6.3.2.2 Methods

In TCustomRVPrint

```
CanSavePDF(827)  
Clear(827)  
GetFirstItemOnPage(827)  
GetPageNo(828)  
UpdatePaletteInfo(828)
```

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`⁽⁷⁷⁸⁾ and `TRVReportHelper.SavePDF` methods.

Currently, PDF saving is implemented only in FireMonkey, if `Skia4Delphi`⁽¹⁵⁷⁾ is available.

6.3.2.2.2 TCustomRVPrint.Clear

Clears memory allocated for printing (in `FormatPages`⁽⁷⁷⁴⁾)

```
procedure Clear;
```

After calling `Clear`, `TRVPrint` will not be able to print and display preview until the next call of `AssignSource`⁽⁷⁷¹⁾ and `FormatPages`⁽⁷⁷⁴⁾ (or `Init`⁽⁸⁰⁹⁾ and `FormatNextPage`⁽⁸⁰⁹⁾ 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 ⁽⁸⁴⁶⁾. Tables can be printed on several pages, and using this information you cannot determine which part of table starts the page.

See also:

- `GetPageNo` ⁽⁸²⁸⁾.

See also properties:

- `PagesCount` ⁽⁸²⁶⁾.

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` ⁽²⁴⁷⁾ of the source TRichView control, or table cell, or `RVData` ⁽²⁴⁷⁾ of a cell inplace editor.

For `TRVPrint` ⁽⁷⁵⁹⁾, the source TRichView control is a control assigned by `AssignSource` ⁽⁷⁷¹⁾. For `TRVReportHelper`, the source TRichView control is `RichView` ⁽⁸⁰⁷⁾ 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 ⁽¹⁶¹⁾, 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` ⁽⁸²⁷⁾.

See also methods of TCustomRichView:

- `GetOffsAfterItem` ⁽³⁰⁸⁾;
- `GetOffsBeforeItem` ⁽³⁰⁹⁾.

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 ²³⁵

TRVPrint uses the same method of working in 256 color mode as its source RichView (see RichView.DoInPaletteMode ²³⁴). Information about source RichView.DoInPaletteMode is updated when executing AssignSource ⁷⁷¹ 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 ⁶²⁵.

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 ⁸³⁸ or RVPrint ⁶²⁸ properties.

See also methods of TRichView:

- UpdatePaletteInfo ³⁶⁸.

6.3.2.3 Events**In TCustomRVPrint**

- OnAfterPrintImage ⁸²⁹
- OnDrawCheckpoint ⁸³⁰
- OnDrawHyperlink ⁸³⁰
- OnPrintComponent ⁸³¹

6.3.2.3.1 TCustomRVPrint.OnAfterPrintImage

Allows custom printing on pictures ¹⁶³ and hot-pictures ¹⁶⁶.

VCL and LCL:**type**

```
TRVAfterPrintImageEvent = procedure(Sender: TCustomRVPrint 822;
  ARVData: TCustomRVData 957; AItem: TCustomRVItemInfo;
  ACanvas: TRVCanvas;
  const ARect: TRVCoordRect 998; APSaD: PRVScreenAndDevice 1024) of object;
```

FireMonkey:**type**

```
TRVAfterPrintImageEvent = procedure(Sender: TCustomRVPrint 822;
  ARVData: TCustomRVData 957; AItem: TCustomRVItemInfo;
  ACanvas: TRVFMXCanvas 961;
  const ARect: TRVCoordRect 998; APSaD: PRVScreenAndDevice 1024) of object;
```

property OnAfterPrintImage: TRVAfterPrintImageEvent;

(introduced in version 20)

Parameters:

ACanvas – canvas for printing.

ARVData – a document containing the item.

Item – 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⁽³⁷⁰⁾

6.3.2.3.2 TCustomRVPrint.OnDrawCheckpoint

Occurs when the component draws (or prints) an item having a *checkpoint*⁽⁸⁷⁾

type

```
TRVDrawCheckpointEvent = procedure (Sender: TCustomRVPrint(822);
  RVData: TCustomRVData(957); ItemNo: Integer; X, Y: TRVCoord(998))
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⁽⁸³⁰⁾.

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(822);
    RVData: TCustomRVData(957); ItemNo: Integer;
    R: TRVCoordRect(998)) 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:

- OnDrawCheckpoint⁸³⁰.

6.3.2.3.4 TCustomRVPrint.OnPrintComponent

Occurs when printing inserted control¹⁶⁸

```
property OnPrintComponent: TRVPrintComponentEvent;
```

```
TRVPrintComponentEvent = procedure(Sender: TCustomRVPrint822;  
  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 others⁹⁸¹. 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 := DrawControl981(PrintMe)
end;

```

In TRVPrint, this event is generated by:

- Print⁷⁷⁶;
- PrintPages⁷⁷⁷;
- MakePreview⁷⁷⁵;
- MakeScaledPreview⁷⁷⁶;
- DrawPreview⁷⁷³;
- when drawing TRVPrintPreview⁶²⁵ component.

6.3.3 TCustomRVPrintPreview

TCustomRVPrintPreview is an ancestor of TRVPrintPreview⁶²⁵ component. This class draws a blank page, introduces mechanisms of page switching and scaling.

Unit [VCL/FMX] CRVPP / fmxCRVPP.

Syntax

```
TCustomRVPrintPreview = class(TRVScroller813)
```

Hierarchy

TObject

TPersistent

TComponent

TControl
TWinControl
TCustomControl
*TRVScroller*⁸¹³

Cursors

Print preview uses two cursors:

- `ZoomInCursor`⁸³⁹
- `ZoomOutCursor`⁸⁴⁰.

Scaling

Scaling is defined by:

- `ZoomMode`⁸⁴⁰ property – mode of scaling;
- `ZoomPercent`⁸⁴¹ property – scaling percent;
- `SetZoom`⁸⁴³ method changes `ZoomPercent`⁸⁴¹;
- `OnZoomChanged`⁸⁴³ event occurs when `ZoomPercent`⁸⁴¹ changes;
- `ClickMode`⁸³⁶ property allows/disallows zooming.

Page switching

`First`⁸⁴¹, `Last`⁸⁴², `Prev`⁸⁴², `Next`⁸⁴² methods, `PageNo`⁸³⁸ property.

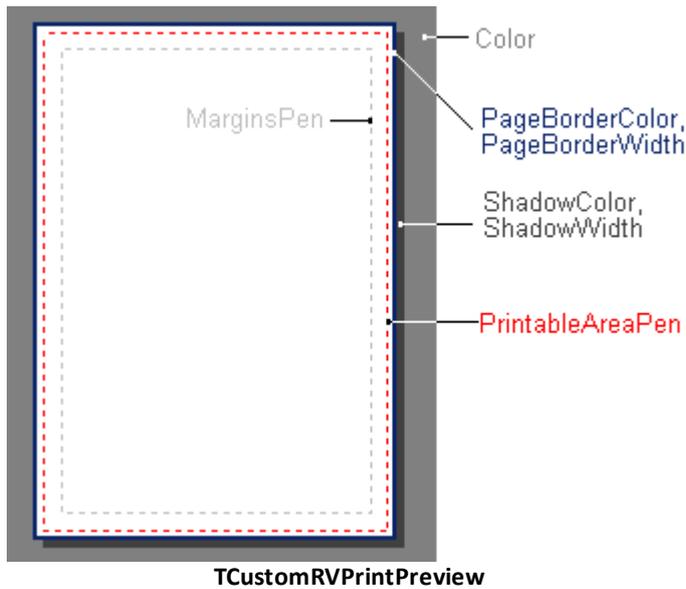
Displaying margins

`MarginsPen/MarginsStroke` property⁸³⁷

Colors and widths

- `Color/Fill`⁸³⁶ – background fill;
- `PageBorderColor`⁸³⁷, `PageBorderWidth`⁸³⁸ define border around the page;
- `ShadowColor`⁸³⁹, `ShadowWidth`⁸³⁹ define shadow;
- `DarkModeUI`⁸³⁷ inverts luminance of colors in user interface of this control.

Scheme



The scheme above is for VCL and LCL version of the component.

FireMonkey version uses `Fill` ⁸³⁶ property instead of `Color` property.

6.3.3.1 Properties

In TCustomRVPrintPreview

- ClickMode ⁸³⁶
- Color ⁸³⁶ [VCL,LCL]
- Fill ⁸³⁶ [FMX]
- MarginsPen ⁸³⁷ [VCL, LCL]
- MarginsStroke ⁸³⁷ [FMX]
- PageBorderColor ⁸³⁷
- PageBorderWidth ⁸³⁸
- PageNo ⁸³⁸
- PrintableAreaPen ⁸³⁸ [VCL, LCL]
- PrintableAreaStroke ⁸³⁸ [FMX]
- ShadowColor ⁸³⁹
- ShadowWidth ⁸³⁹
- ZoomInCursor ⁸³⁹
- ZoomMode ⁸⁴⁰
- ZoomOutCursor ⁸⁴⁰
- ZoomPercent ⁸⁴¹

Derived from TRVScroller ⁸¹³

- BorderStyle ⁸¹⁶ [VCL]
- ▶ HScrollMax ⁸¹⁶
- HScrollPos ⁸¹⁶
- HScrollVisible ⁸¹⁶ [VCL,LCL]

- ▶ InplaceEditor⁸¹⁷ (not used in this component)
- NoVScroll⁸¹⁷
- Tracking⁸¹⁸
- UseXPThemes⁸¹⁸ [VCL,LCL]
- ▶ VScrollMax⁸¹⁸
- VScrollPos⁸¹⁹
- VScrollVisible⁸¹⁹ [VCL,LCL]
- WheelStep⁸¹⁹

Derived from TCustomControl [VCL/LCL] or TCustomScrollBar [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
- ShowScrollBar [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
<i>rvcmNone</i>	Clicking does nothing, zoom-in ⁽⁸³⁹⁾ and zoom-out ⁽⁸⁴⁰⁾ cursors are not used.
<i>rvcmSwitchZoom</i>	Clicking switches between 100% and custom zoom.

Default value:

rvcmSwitchZoom

6.3.3.1.2 TCustomRVPrintPreview.Color, Fill

Background fill.

VCL:

property Color: TRVColor⁽⁹⁹⁶⁾;

(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:

rvcIBtnShadow⁽¹⁰³⁸⁾

See also:

- DarkModeUI⁽⁸³⁷⁾
- Picture at the bottom of TCustomRVPrintPreview⁽⁸³²⁾ 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, RVPrint⁶²⁸.DarkMode⁸²³ 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/PrintableAreaStroke⁸³⁸.

See also properties of TRVPrint⁷⁵⁹:

- Margins⁷⁶⁸,

See also:

- Picture in TRVPrint topic⁷⁵⁹.

6.3.3.1.5 TCustomRVPrintPreview.PageBorderColor

Color of frame around the page.

```
property PageBorderColor: TRVColor996;
```

(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` ⁽¹⁰³⁸⁾

See also:

- `DarkModeUI` ⁽⁸³⁷⁾
- Picture at the bottom of `TCustomRVPrintPreview` ⁽⁸³²⁾ topic.

6.3.3.1.6 TCustomRVPrintPreview.PageBorderWidth

Width of frame around the page, pixels

property `PageBorderWidth`: `TRVPixel96Length` ⁽¹⁰¹⁸⁾;

(introduced in v1.8)

Default value:

2

See also:

- `DarkModeUI` ⁽⁸³⁷⁾
- Picture at the bottom of `TCustomRVPrintPreview` ⁽⁸³²⁾ topic.

6.3.3.1.7 TCustomRVPrintPreview.PageNo

Index of displayed page (from 1)

property `PageNo`: `Integer`;

See also methods:

- `Prev` ⁽⁸⁴²⁾;
- `Next` ⁽⁸⁴²⁾;
- `First` ⁽⁸⁴¹⁾;
- `Last` ⁽⁸⁴²⁾.

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⁽⁸³⁷⁾.

6.3.3.1.9 TCustomRVPrintPreview.ShadowColor

Color of the page shadow.

property ShadowColor: TRVColor⁽⁹⁹⁶⁾;

(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⁽¹⁰³⁸⁾

See also:

- DarkModeUI⁽⁸³⁷⁾
- Picture at the bottom of TCustomRVPrintPreview⁽⁸³²⁾ topic.

6.3.3.1.10 TCustomRVPrintPreview.ShadowWidth

Width of the page shadow, pixels.

property ShadowWidth: TRVPixel96Length⁽¹⁰¹⁸⁾;

(introduced in v1.8)

Default value:

20

See also:

- Picture at the bottom of TCustomRVPrintPreview⁽⁸³²⁾ 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⁽⁸⁴¹⁾<>100%, clicking on component changes ZoomPercent⁽⁸⁴¹⁾ to 100%.

If ZoomPercent⁽⁸⁴¹⁾=100%, clicking on component changes ZoomPercent⁽⁸⁴¹⁾ to previous scale.

Default value:*crRVZoomIn* (102)*(this cursor is registered automatically)***See also properties:**

- *ZoomOutCursor*⁽⁸⁴⁰⁾;
- *ZoomPercent*⁽⁸⁴¹⁾.

See also:

- RichView cursors⁽¹⁴²⁾.

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
<i>rvzmFullPage</i>	Changing <i>ZoomPercent</i> ⁽⁸⁴¹⁾ to display full page, keeping the full-page view during resizing (changing <i>ZoomPercent</i> ⁽⁸⁴¹⁾ when needed)
<i>rvzmPageWidth</i>	Changing <i>ZoomPercent</i> ⁽⁸⁴¹⁾ to fit width of page, keeping the page-width view during resizing (changing <i>ZoomPercent</i> ⁽⁸⁴¹⁾ when needed)
<i>rvzmCustom</i>	Scaling according to the <i>ZoomPercent</i> ⁽⁸⁴¹⁾ 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*⁽⁸⁴¹⁾ <>100%, clicking on component changes *ZoomPercent*⁽⁸⁴¹⁾ to 100%.If *ZoomPercent*⁽⁸⁴¹⁾ =100%, clicking on component changes *ZoomPercent*⁽⁸⁴¹⁾ to previous scale.**Default value:***crRVZoomOut* (103)*(this cursor is registered automatically)***See also properties:**

- *ZoomInCursor*⁽⁸³⁹⁾;

- ZoomPercent⁸⁴¹.

See also:

- RichView cursors¹⁴².

6.3.3.1.14 TCustomRVPrintPreview.ZoomPercent

Scale of preview, %

```
property ZoomPercent: TRVZoomPercent1035;
```

Although assigning to this property is possible, the preferable way to change preview scale is calling SetZoom⁸⁴³ method.

This property is changed automatically when resizing the component, if ZoomMode⁸⁴⁰ = *rvzmFullPage* or *rvzmPageWidth*.

It's recommended to keep this property in 10%–500% range.

See also methods:

- SetZoom⁸⁴³.

See also properties:

- ZoomMode⁸⁴⁰.

See also events:

- OnZoomChanged⁸⁴³.

6.3.3.2 Methods

In TCustomRVPrintPreview

Create⁸⁴¹
First⁸⁴¹
Last⁸⁴²
Next⁸⁴²
Prev⁸⁴²
SetZoom⁸⁴³

Derived from TRVScroller⁸¹³

ScrollTo⁸²⁰ (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 TRVPrintPreview⁶²⁵ components instead.

6.3.3.2.2 TCustomRVPrintPreview.First

Switches the current page (PageNo⁸³⁸ property) to the first page.

```
procedure First;
```

See also properties:

- PageNo⁸³⁸ .

See also methods:

- Prev⁸⁴² ;
- Next⁸⁴² ;
- Last⁸⁴² .

6.3.3.2.3 TCustomRVPrintPreview.Last

Switches the current page (PageNo⁸³⁸ property) to the last page.

procedure Last ;

See also properties:

- PageNo⁸³⁸ ;
- TRVPrintPreview.RVPrint⁶²⁸ .PagesCount⁸²⁶ .

See also methods:

- Prev⁸⁴² ;
- Next⁸⁴² ;
- First⁸⁴¹ .

6.3.3.2.4 TCustomRVPrintPreview.Next

Switches the current page (PageNo⁸³⁸ property) to the next page, if possible.

procedure Next ;

See also properties:

- PageNo⁸³⁸ ;
- TRVPrintPreview.RVPrint⁶²⁸ .PagesCount⁸²⁶ .

See also methods:

- Prev⁸⁴² ;
- First⁸⁴¹ ;
- Last⁸⁴² .

6.3.3.2.5 TCustomRVPrintPreview.Prev

Switches the current page (PageNo⁸³⁸ property) to the previous page, if possible.

procedure Prev ;

See also properties:

- PageNo⁸³⁸ .

See also methods:

- Next⁸⁴² ;
- First⁸⁴¹ ;
- Last⁸⁴² .

6.3.3.2.6 TCustomRVPrintPreview.SetZoom

Changes ZoomMode⁽⁸⁴⁰⁾ to *rvzmCustom* and ZoomPercent⁽⁸⁴¹⁾ to **Percent**.

```
procedure SetZoom(Percent: Integer(1035));
```

Calling this method has an immediate effect.

It's recommended to keep **Percent** in 10%–500% range.

See also properties:

- ZoomMode⁽⁸⁴⁰⁾;
- ZoomPercent⁽⁸⁴¹⁾.

6.3.3.3 Events

In TCustomRVPrintPreview

- OnZoomChanged⁽⁸⁴³⁾

Derived from TRVScroller⁽⁸¹³⁾

- OnHScrolled⁽⁸²¹⁾
- OnVScrolled⁽⁸²¹⁾

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⁽⁸⁴¹⁾ changes.

```
property OnZoomChanged: TNotifyEvent;
```

7 Item types

TCustomRVItemInfo is a basic type for all items ⁽¹⁵⁸⁾ 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 ⁽²¹⁰⁾ and TRichViewEdit ⁽⁴⁶¹⁾ instead of working with the item classes directly, so details of implementation are hidden from programmers. The exceptions are:

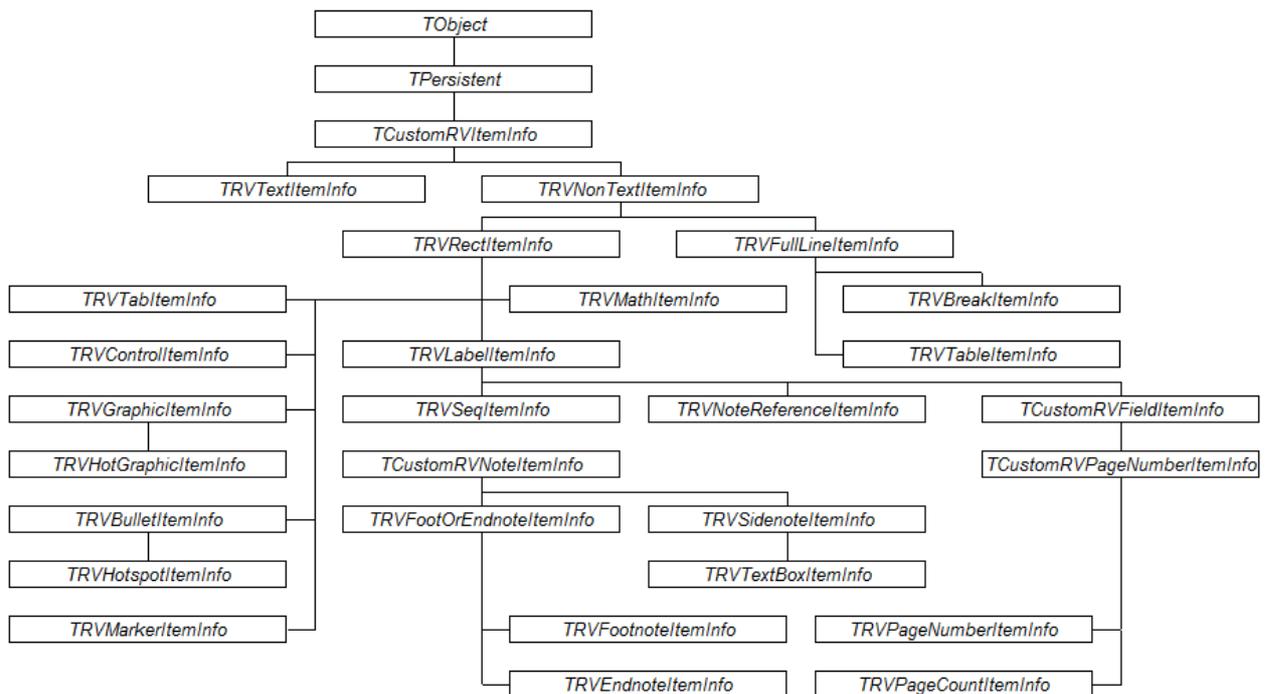
- TRVTableItemInfo ⁽⁸⁴⁶⁾;
- TRVLabelItemInfo ⁽⁹¹²⁾ and classes, inherited from it.

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 ⁽¹⁶¹⁾ .
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 ⁽¹⁶²⁾ .
TRVControlItemInfo	RVItem	Item containing Delphi control ⁽¹⁶⁸⁾ .
TRVGraphicItemInfo	RVItem	Picture ⁽¹⁶³⁾ .
TRVHotGraphicItemInfo	RVItem	<i>Hot-picture</i> ⁽¹⁶⁶⁾ (picture-hyperlink).
TRVBulletItemInfo	RVItem	<i>Bullet</i> ⁽¹⁷⁰⁾ (image from TImageList).
TRVHotspotItemInfo	RVItem	<i>Hotspot</i> ⁽¹⁷²⁾ (<i>bullet</i> -hyperlink).
TRVMarkerItemInfo	RVMarker	Paragraph marker ⁽¹⁷⁴⁾ .
TRVTableItemInfo ⁽⁸⁴⁶⁾	RVTable	Table ⁽¹⁷³⁾ .
TRVBreakItemInfo	RVItem	<i>Break</i> ⁽¹⁶⁷⁾ (horizontal line).
TRVLabelItemInfo ⁽⁹¹²⁾	RVLabelItem	Label ⁽¹⁷⁶⁾ (non-text item looking like text).
TRVSeqItemInfo ⁽⁹²¹⁾	RVSeqItem	Numbered sequence ⁽¹⁷⁷⁾ .
TCustomRVNoteItemInfo ⁽⁹²⁵⁾	RVNote	Basic class for notes and text boxes.
TRVFootOrEndnoteItemInfo ⁽⁹²⁵⁾	RVNote	Basic class for footnotes and endnotes

Class	Unit	Meaning
TRVEndnoteItemInfo ⁽⁹²⁸⁾	RVNote	Endnote ⁽¹⁷⁸⁾ .
TRVFootnoteItemInfo ⁽⁹³⁰⁾	RVNote	Footnote ⁽¹⁸⁰⁾ .
TRVSidenoteItemInfo ⁽⁹³³⁾	RVSidenote	Sidenote ⁽¹⁸¹⁾
TRVTextBoxItemInfo ⁽⁹⁴⁷⁾	RVSidenote	Text box ⁽¹⁸³⁾
TRVNoteReferenceItemInfo ⁽⁹⁴⁹⁾	RVNote	Reference to the parent note ⁽¹⁸⁴⁾
TRVPageNumberItemInfo ⁽⁹⁵³⁾	RVFieldItems	"Page number" field ⁽¹⁸⁵⁾
TRVPageCountItemInfo ⁽⁹⁵⁴⁾	RVFieldItems	"Page count" field ⁽¹⁸⁶⁾
TRVMathItemInfo ⁽⁹¹⁷⁾	RVMathItem	Mathematical expression ⁽¹⁸⁷⁾

Hierarchy



Other Types

```
TCustomRVItemInfoClass = class of TCustomRVItemInfoClass;
```

7.1 Table - TRVTableItemInfo

TRVTableItemInfo is a class representing table ⁽¹⁷³⁾ in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem ⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem ⁽⁵²²⁾). Style ⁽³⁰²⁾ of this item type: *rvsTable* (-60)

Unit RVTable.

Syntax

```
TRVTableItemInfo = class (TRVFullLineItemInfo)
```

Hierarchy

TObject

TPersistent

TCustomRVItemInfo ⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVFullLineItemInfo

Overview

See overview topics about tables ⁽¹⁹⁰⁾.

Properties

The most of properties of this class are published. These properties can be saved/loaded in RVF ⁽¹²⁴⁾ files or streams.

The main property of TRVTableItemInfo is Cells ⁽⁸⁵⁷⁾. The number of rows is RowCount ⁽⁸⁶⁴⁾, number of columns is ColCount ⁽⁸⁵⁸⁾.

Properties of rows are available via Rows ⁽⁸⁶⁴⁾ property.

Width of table can be defined using BestWidth ⁽⁸⁵⁰⁾ property.

Properties defining border around the table:

- BorderWidth ⁽⁸⁵³⁾;
- BorderStyle ⁽⁸⁵²⁾;
- BorderColor ⁽⁸⁵¹⁾;
- BorderLightColor ⁽⁸⁵²⁾;
- VisibleBorders ⁽⁸⁶⁵⁾;
- also BorderHSpacing ⁽⁸⁵¹⁾, BorderVSpacing ⁽⁸⁵³⁾.

Properties defining default border around cells:

- CellBorderWidth ⁽⁸⁵⁵⁾;
- CellBorderStyle ⁽⁸⁵⁴⁾;
- CellBorderColor ⁽⁸⁵³⁾;
- CellBorderLightColor ⁽⁸⁵⁴⁾.

Properties defining default colors of cells in rows and columns:

- HeadingRowColor ⁽⁸⁶⁷⁾, LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor

Properties defining spacing between cells:

- CellHSpacing⁽⁸⁵⁶⁾;
- CellVSpacing⁽⁸⁵⁷⁾;
- BorderHSpacing⁽⁸⁵¹⁾;
- BorderVSpacing⁽⁸⁵³⁾;
- also CellHPadding⁽⁸⁵⁵⁾, CellVPadding⁽⁸⁵⁷⁾.

Background properties:

- Color⁽⁸⁵⁸⁾;
- Opacity⁽⁸⁶⁰⁾;
- BackgroundImage⁽⁸⁴⁹⁾;
- BackgroundStyle⁽⁸⁵⁰⁾;
- BackgroundImageFileName⁽⁸⁵⁰⁾.

Border colors:

- BorderColor⁽⁸⁵¹⁾;
- BorderLightColor⁽⁸⁵²⁾;
- CellBorderColor⁽⁸⁵³⁾;
- CellBorderLightColor⁽⁸⁵⁴⁾.

Rules:

- HRuleWidth⁽⁸⁶⁰⁾;
- HRuleColor⁽⁸⁶⁰⁾;
- HOutermostRule⁽⁸⁵⁹⁾;
- VRuleWidth⁽⁸⁶⁷⁾;
- VRuleColor⁽⁸⁶⁶⁾;
- VOutermostRule⁽⁸⁶⁶⁾.

Export:

- TextColSeparator⁽⁸⁶⁵⁾;
- TextRowSeparator⁽⁸⁶⁵⁾;
- BackgroundImageFileName⁽⁸⁵⁰⁾.

Other properties:

- Options⁽⁸⁶¹⁾;
- PrintOptions⁽⁸⁶³⁾;
- HeadingRowCount⁽⁸⁵⁹⁾.

Methods

See operations on tables⁽¹⁹⁹⁾.

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⁽¹²⁴⁾ files or streams. Reassign them when tables are loaded, use TRichView.OnItemAction⁽³⁸⁰⁾ event.

Events:

- OnCellEditing⁽⁸⁹²⁾;

- OnDrawBorder⁸⁹³.

Related

- Drawing table grid⁶⁴⁰;
- New row auto insertion¹⁰⁴⁹.

7.1.1 Properties

In TRVTableItemInfo

- BackgroundImage⁸⁴⁹
- BackgroundImageFileName⁸⁵⁰
- BackgroundStyle⁸⁵⁰
- BestWidth⁸⁵⁰
- BorderColor⁸⁵¹
- BorderLightColor⁸⁵²
- BorderHSpacing⁸⁵¹
- BorderVSpacing⁸⁵³
- BorderStyle⁸⁵²
- BorderWidth⁸⁵³
- CellBorderColor⁸⁵³
- CellBorderLightColor⁸⁵⁴
- CellBorderStyle⁸⁵⁴
- CellBorderWidth⁸⁵⁵
- CellHPadding⁸⁵⁵
- CellHSpacing⁸⁵⁶
- CellOverrideColor⁸⁵⁶
- CellPadding⁸⁵⁶
- Cells⁸⁵⁷
- CellVPadding⁸⁵⁷
- CellVSpacing⁸⁵⁷
- ▶ ColCount⁸⁵⁸
- Color⁸⁵⁸
- EvenColumnsColor⁸⁶⁷
- EvenRowsColor⁸⁶⁷
- FirstColumnColor⁸⁶⁷
- HeadingRowColor⁸⁶⁷
- HeadingRowCount⁸⁵⁹
- HOutermostRule⁸⁵⁹
- HRuleColor⁸⁶⁰
- HRuleWidth⁸⁶⁰
- LastColumnColor⁸⁶⁷
- LastRowColor⁸⁶⁷
- OddColumnsColor⁸⁶⁷
- OddRowsColor⁸⁶⁷
- Opacity⁸⁶⁰
- Options⁸⁶¹
- PrintOptions⁸⁶³

- ▶ RowCount⁸⁶⁴
- Rows⁸⁶⁴
- TextColSeparator⁸⁶⁵
- TextRowSeparator⁸⁶⁵
- VisibleBorders⁸⁶⁵
- VOutermostRule⁸⁶⁶
- VRuleColor⁸⁶⁶
- VRuleWidth⁸⁶⁷

7.1.1.1 TRVTableItemInfo.BackgroundImage

Background image for the table.

```
property BackgroundImage: TRVGraphic970;
```

(introduced in v1.8)

Position of the image is defined by the BackgroundStyle⁸⁵² 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 RichViewMaxPictureCount¹⁰⁴⁶).

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:

- BackgroundStyle⁸⁵⁰;
- Color⁸⁵⁸.

See also properties of table cell:

- BackgroundImage⁸⁹⁶;
- BackgroundStyle⁸⁹⁷.

See also properties of TRichView:

- BackgroundBitmap²²²;
- BackgroundStyle²²³.

7.1.1.2 TRVTableItemInfo.BackgroundImageFileName

A string storing the file name associated with BackgroundImage⁽⁸⁴⁹⁾.

property BackgroundImageFileName: TRVUnicodeString⁽¹⁰³²⁾;

(introduced in v1.9; changed in version 18)

If *rvhtmlsioUseItemImageFileNames* in HTMLSaveProperties⁽²³⁷⁾.ImageOptions⁽⁴³⁴⁾, 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⁽¹⁰⁰⁵⁾ *rvsplImageFileName*.

Assignment to this property cannot be undone by the user⁽¹¹⁵⁾; to make an undoable assignment, use *rvsplImageFileName* property.

See also properties of table cell:

- BackgroundImageFileName⁽⁸⁹⁷⁾.

7.1.1.3 TRVTableItemInfo.BackgroundStyle

Defines position of BackgroundImage⁽⁸⁴⁹⁾.

property BackgroundStyle: TRVItemBackgroundStyle⁽¹⁰¹³⁾;

(introduced in v1.8)

See also:

- Undo of assignments to table properties.

See also properties:

- BackgroundImage⁽⁸⁴⁹⁾;
- Color⁽⁸⁵⁸⁾.

See also properties of table cell:

- BackgroundStyle⁽⁸⁹⁷⁾;
- BackgroundImage⁽⁸⁹⁶⁾.

See also properties of TRichView:

- BackgroundStyle⁽²²³⁾;
- BackgroundBitmap⁽²²²⁾.

7.1.1.4 TRVTableItemInfo.BestWidth

Preferred width for the table.

property BestWidth: TRVHTMLLength⁽¹⁰¹²⁾;

This is a preferred width for the table, measured either in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ 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⁽¹⁰²⁷⁾ (have at least one cell with BestWidth⁽⁸⁹⁸⁾ defined in TRVStyleUnits⁽¹⁰²⁷⁾, and do not have cells with BestWidth⁽⁸⁹⁸⁾ 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⁽²⁰⁵⁾;
- Undo of assignments to table properties.

See also properties of table cell:

- BestWidth⁽⁸⁹⁸⁾;
- BestHeight⁽⁸⁹⁸⁾.

7.1.1.5 TRVTableItemInfo.BorderColor

Defines color of the table border.

property BorderColor: TRVColor⁽⁹⁹⁶⁾

This property defines either color of flat border or "dark" color of 3d border around the table.

See BorderStyle⁽⁸⁵²⁾ for details.

This property corresponds to HTML <table bordercolor> or <table bordercolordark> attributes (specific to Microsoft Internet Explorer).

Default value:

rvclWindowText⁽¹⁰³⁸⁾

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties

- BorderLightColor⁽⁸⁵²⁾;
- CellBorderColor⁽⁸⁵³⁾;
- CellBorderLightColor⁽⁸⁵⁴⁾.

7.1.1.6 TRVTableItemInfo.BorderHSpacing

Horizontal spacing between the table border and the outermost cells.

property BorderHSpacing: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

BorderHSpacing be saved in HTML+CSS if:

- BorderHSpacing=CellHSpacing⁽⁸⁵⁶⁾, or
- BorderHSpacing>=CellHSpacing⁽⁸⁵⁶⁾ and CellHSpacing⁽⁸⁵⁶⁾>=0

Default value:

2

Note: this default value assumes that RVStyle.Units⁽⁶⁵⁵⁾=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables ⁽¹⁹³⁾ (with pictures).

See also properties:

- `BorderVSpacing` ⁽⁸⁵³⁾;
- `CellHSpacing` ⁽⁸⁵⁶⁾;
- `CellVSpacing` ⁽⁸⁵⁷⁾.

7.1.1.7 TRVTableItemInfo.BorderLightColor

This property defines "light" color of 3d border around the table.

```
property BorderLightColor: TRVColor (996) ;
```

See `BorderStyle` ⁽⁸⁵²⁾ for details.

This property corresponds to HTML `<table bordercolorlight>` attribute (specific to Microsoft Internet Explorer).

Default value:

`rvclBtnHighlight` ⁽¹⁰³⁸⁾

See also:

- Undo of assignments to table properties;
- Colors and layout of tables ⁽¹⁹³⁾ (with pictures).

See also properties:

- `BorderColor` ⁽⁸⁵¹⁾;
- `CellBorderColor` ⁽⁸⁵³⁾;
- `CellBorderLightColor` ⁽⁸⁵⁴⁾.

7.1.1.8 TRVTableItemInfo.BorderStyle

This property defines style of border around the table.

```
property BorderStyle: TRVTableBorderStyle (1029) ;
```

If you want to hide table border, set `BorderWidth` ⁽⁸⁵³⁾ = 0.

Default value:

`rvtbRaised`

See also:

- Undo of assignments to table properties;
- Colors and layout of tables ⁽¹⁹³⁾ (with pictures).

See also properties:

- `CellBorderStyle` ⁽⁸⁵⁴⁾;
- `BorderColor` ⁽⁸⁵¹⁾;
- `BorderLightColor` ⁽⁸⁵²⁾;
- `BorderWidth` ⁽⁸⁵³⁾;
- `VisibleBorders` ⁽⁸⁶⁵⁾.

7.1.1.9 TRVTableItemInfo.BorderVSpacing

Vertical spacing between border and the outermost cells.

property BorderVSpacing: TRVStyleLength¹⁰²⁷;

This value is measured in Units⁶⁵⁵ of the linked TRVStyle⁶³⁰ component.

BorderVSpacing be saved in HTML+CSS if:

- BorderVSpacing=CellVSpacing⁸⁵⁷, or
- BorderVSpacing>=CellVSpacing⁸⁵⁷ and CellVSpacing⁸⁵⁷>=0

Default value:

2

Note: this default value assumes that RVStyle.Units⁶⁵⁵=rvstuPixels, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables¹⁹³ (with pictures).

See also properties:

- BorderHSpacing⁸⁵¹;
- CellHSpacing⁸⁵⁶;
- CellVSpacing⁸⁵⁷.

7.1.1.10 TRVTableItemInfo.BorderWidth

Width of border around the table.

property BorderWidth: TRVStyleLength¹⁰²⁷;

This value is measured in Units⁶⁵⁵ of the linked TRVStyle⁶³⁰ component.

Set to 0 to hide border.

This property corresponds to HTML <table border>.

Default value:

0

See also:

- Undo of assignments to table properties;
- Colors and layout of tables¹⁹³ (with pictures).

See also properties:

- CellBorderWidth⁸⁵⁵;
- VisibleBorders⁸⁶⁵.

7.1.1.11 TRVTableItemInfo.CellBorderColor

Defines default border color for all table cells.

property CellBorderColor: TRVColor⁹⁹⁶

This property defines either default color of flat border or default "dark" color of 3d border around the table cells.

See CellBorderStyle⁸⁵⁴ for details.

This value can be overridden with `BorderColor`⁽⁸⁹⁹⁾ of cell.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

Default value:

`rvclWindowText`⁽¹⁰³⁸⁾

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- `CellBorderLightColor`⁽⁸⁵⁴⁾;
- `BorderLightColor`⁽⁸⁵²⁾;
- `BorderColor`⁽⁸⁵¹⁾.

See also properties of table cell:

- `BorderColor`⁽⁸⁹⁹⁾;
- `BorderLightColor`⁽⁸⁹⁹⁾.

7.1.1.12 TRVTableItemInfo.CellBorderLightColor

Defines default "light" color of 3d border for all table cells.

property `CellBorderLightColor`: `TRVColor`⁽⁹⁹⁶⁾;

See `CellBorderStyle`⁽⁸⁵⁴⁾ for details. To hide cell borders, set `CellBorderWidth`⁽⁸⁵⁵⁾ = 0.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

This value can be overridden with `BorderLightColor`⁽⁸⁹⁹⁾ of cell.

Default value:

`rvclBtnHighlight`⁽¹⁰³⁸⁾

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- `CellBorderColor`⁽⁸⁵³⁾;
- `BorderLightColor`⁽⁸⁵²⁾;
- `BorderColor`⁽⁸⁵¹⁾.

See also properties of table cell:

- `BorderColor`⁽⁸⁹⁹⁾;
- `BorderLightColor`⁽⁸⁹⁹⁾.

7.1.1.13 TRVTableItemInfo.CellBorderStyle

Style of border around the table cells

property `BorderStyle`: `TRVTableBorderStyle`⁽¹⁰²⁹⁾;

To hide cell borders, set `CellBorderWidth`⁽⁸⁵⁵⁾ = 0.

Default value:

`rvtbLowered`

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- BorderStyle⁽⁸⁵²⁾;
- CellBorderWidth⁽⁸⁵⁵⁾;
- CellBorderColor⁽⁸⁵³⁾;
- CellBorderLightColor⁽⁸⁵⁴⁾.

7.1.1.14 TRVTableItemInfo.CellBorderWidth

Width of border around the cells of table, pixels

property CellBorderWidth: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

Set to 0 to hide border.

If value of this property <> BorderWidth⁽⁸⁵³⁾, 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 properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- BorderWidth⁽⁸⁵³⁾;
- CellBorderStyle⁽⁸⁵⁴⁾;
- CellBorderColor⁽⁸⁵³⁾;
- CellBorderLightColor⁽⁸⁵⁴⁾.

7.1.1.15 TRVTableItemInfo.CellHPadding

Horizontal distance from the cell content to the cell border.

property CellHPadding: TRVStyleLength⁽¹⁰²⁷⁾;

(introduced in v10)

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

This value affects all cells in the table.

If the cell is rotated⁽⁹⁰⁴⁾, CellHPadding and CellVPadding⁽⁸⁵⁷⁾ are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units⁽⁶⁵⁵⁾ = *rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- CellVPadding⁽⁸⁵⁷⁾;
- CellPadding⁽⁸⁵⁶⁾.

7.1.1.16 TRVTableItemInfo.CellHSpacing

Horizontal spacing between cells.

property CellHSpacing: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

This value can be negative. For example, you can assign CellHSpacing=-CellBorderWidth⁽⁸⁵⁵⁾ to collapse cell borders horizontally.

Default value:

2

Note: this default value assumes that RVStyle.Units⁽⁶⁵⁵⁾=rvstuPixels, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- CellVSpacing⁽⁸⁵⁷⁾;
- BorderHSpacing⁽⁸⁵¹⁾;
- BorderVSpacing⁽⁸⁵³⁾.

7.1.1.17 TRVTableItemInfo.CellOverrideColor

A color to override cell background colors.

property CellOverrideColor: TRVColor⁽⁹⁹⁶⁾;

(introduced in v17)

If CellOverrideColor<>rvclNone⁽¹⁰³⁸⁾, all cells are drawn using this color. Colors of individual cells⁽⁹⁰⁰⁾ and colors of bands of rows and columns⁽⁸⁶⁷⁾ are ignored.

Default value:

rvclNone⁽¹⁰³⁸⁾

7.1.1.18 TRVTableItemInfo.CellPadding

Distance between cell content and cell border.

property CellPadding: TRVStyleLength⁽¹⁰²⁷⁾;

This property is maintained for backward compatibility. Assigning to this property assigns values to CellHPadding⁽⁸⁵⁵⁾ and CellVPadding⁽⁸⁵⁷⁾. Reading this property returns Round((CellHPadding⁽⁸⁵⁵⁾+CellVPadding⁽⁸⁵⁷⁾)/2).

7.1.1.19 TRVTableItemInfo.Cells

Provides indexed access to all cells in the table.

property Cells[Row, Col: Integer]: TRVTableCellData⁽⁸⁹⁵⁾;

This is the key property of table.

Row can be in range from 0 to RowCount⁽⁸⁶⁴⁾-1. **Col** can be in range from 0 to ColCount⁽⁸⁵⁸⁾-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⁽⁸⁶⁴⁾.

See also:

- Table Cells Overview⁽¹⁹²⁾.

7.1.1.20 TRVTableItemInfo.CellVPadding

Vertical distance from the cell content to the cell border.

property CellVPadding: TRVStyleLength⁽¹⁰²⁷⁾;

(introduced in v10)

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

This value affects all cells in the table.

If the cell is rotated⁽⁹⁰⁴⁾, CellHPadding⁽⁸⁵⁵⁾ and CellVPadding are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units⁽⁶⁵⁵⁾=rvstuPixels, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- CellHPadding⁽⁸⁵⁵⁾;
- CellPadding⁽⁸⁵⁶⁾.

7.1.1.21 TRVTableItemInfo.CellVSpacing

Vertical spacing between cells.

property CellVSpacing: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

This value can be negative. For example, you can assign CellVSpacing=-CellBorderWidth⁽⁸⁵⁵⁾ to collapse cell borders vertically.

Default value:

2

Note: this default value assumes that `RVStyle.Units(655) = rvstuPixels`, otherwise it could be too small.

See also:

- Undo of assignments to table properties;
- Colors and layout of tables⁽¹⁹³⁾ (with pictures).

See also properties:

- `CellHSpacing(856)`;
- `BorderHSpacing(851)`;
- `BorderVSpacing(853)`.

7.1.1.22 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⁽⁸⁶⁷⁾.

The heading (or the first row) and the first column may be excluded from bands.

Default values

1

7.1.1.23 TRVTableItemInfo.ColCount

Returns the number of columns in the table.

```
property ColCount: Integer
```

(introduced in v10)

If `RowCount(864) > 0`, this property returns `Rows(864)[0].Count` (all rows have the same number of cells).
If `RowCount(864) = 0`, this property returns 0.

See also properties:

- `Cells(857)`;
- `Rows(864)`.

7.1.1.24 TRVTableItemInfo.Color

Background color of the table

```
property Color: TRVColor(996);
```

Set to `rvclNone(1038)` for transparent background. You can specify the background `Opacity(860)`.

If the table has a background image⁽⁸⁴⁹⁾, this image is drawn on top of color filling.

Default value:

*rvclWindow*¹⁰³⁸

See also:

- Undo of assignments to table properties;
- Colors and layout of tables¹⁹³ (with pictures).

7.1.1.25 TRVTableItemInfo.HeadingRowCount

Specifies the number of rows that will be repeated on each printer page with this table.

property HeadingRowCount: Integer;

(introduced in version 1.7)

If these rows overlap next rows (due to cell merging), this property is ignored on pagination.

A special color (HeadingRowColor⁸⁶⁷) 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:

- PrintOptions⁸⁶³,
- ColBandSize, RowBandSize⁸⁵⁸

7.1.1.26 TRVTableItemInfo.HOutermostRule

Specifies if the horizontal rules should be drawn between the table border and the outermost cells.

property HOutermostRule: Boolean;

These rules can be visible only if other rules are visible (HRuleWidth⁸⁶⁰>0)

Note about rules.

Default value:

False

See also:

- Undo of assignments to table properties;
- Table rules¹⁹⁴ (with pictures).

See also properties:

- VOutermostRule⁸⁶⁶;
- HRuleWidth⁸⁶⁰;
- HRuleColor⁸⁶⁰.

7.1.1.27 TRVTableItemInfo.HRuleColor

Specifies the color of horizontal rules (horizontal lines between cells)

property HRuleColor: TRVColor⁹⁹⁶;

Horizontal rules are visible only if HRuleWidth⁸⁶⁰>0

Note about rules.

Default value:

*rvclWindowText*¹⁰³⁸

See also:

- Undo of assignments to table properties;
- Table rules¹⁹⁴ (with pictures).

See also properties:

- VRuleColor⁸⁶⁶;
- HRuleWidth⁸⁶⁰;
- HOutermostRule⁸⁵⁹.

7.1.1.28 TRVTableItemInfo.HRuleWidth

Specifies the width of horizontal rules (horizontal lines between cells)

property HRuleWidth: TRVStyleLength¹⁰²⁷;

This value is measured in Units⁶⁵⁵ of the linked TRVStyle⁶³⁰ 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 rules¹⁹⁴ (with pictures).

See also properties

- VRuleWidth⁸⁶⁷;
- HRuleColor⁸⁶⁰;
- HOutermostRule⁸⁵⁹.

7.1.1.29 TRVTableItemInfo.Opacity

Opacity of table background.

property Opacity: TRVOpacity¹⁰¹⁵;

(introduced in version 16)

This value is used if Color⁸⁵⁸<>*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 ⁽¹⁹³⁾.

7.1.1.30 TRVTableItemInfo.Options

Specifies options for the table.

type

```
TRVTableOption = (rvtoEditing,
  rvtoRowSizing, rvtoColSizing,
  rvtoRowSelect, rvtoColSelect,
  rvtoNoCellSelect,
  rvtoRTFSaveCellPixelBestWidth,
  rvtoHideGridLines,
  rvtoCellBelowBorders,
  rvtoOverlappingCorners,
  rvtoRTFAllowAutofit,
  rvtoIgnoreContentWidth,
  rvtoIgnoreContentHeight);
TRVTableOptions = set of TRVTableOption;
```

property Options: TRVTableOptions;**Default value**[*rvtoEditing, rvtoRowSizing, rvtoColSizing, rvtoRowSelect, rvtoColSelect*]**Options for Editing**

Option	Meaning
<i>rvtoEditing</i>	If set (default), users can edit content of cells using inplace editor. (table must be inserted in TRichViewEdit ⁽⁴⁶¹⁾ or TDBRichViewEdit ⁽⁶⁰⁶⁾)

Options for Resizing and Selection

Option	Meaning
<i>rvtoRowSizing</i> <i>rvtoColSizing</i>	If set (default), users can resize individual rows/columns with the mouse. (table must be inserted in TRichViewEdit ⁽⁴⁶¹⁾ or TDBRichViewEdit ⁽⁶⁰⁶⁾)
<i>rvtoRowSelect</i> , <i>rvtoColSelect</i>	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)

Option	Meaning
	The options have no effect if the table is inserted in RichView with <i>rvAllowSelection</i> excluded from Options ⁽²⁴⁰⁾ .
<i>rvtoNoCellSelect</i>	If set, multicell selection by mouse is not allowed (remove also <i>rvtoRowSelect</i> and <i>rvtoColSelect</i> to disallow making multicell selection completely.

Layout Options

Option	Meaning
<i>rvtolgnoreContentWidth</i>	If set, contents of cells are completely ignored when calculating widths of table columns. Wide pictures, controls, etc. are clipped.
<i>rvtolgnoreContentHeight</i>	If set, heights of rows are calculated basing only on BestHeight ⁽⁸⁹⁸⁾ properties of cells. But (unlike <i>rvtolgnoreContentWidth</i>), 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 ⁽⁹⁰¹⁾ property instead.

Visual Options

Option	Meaning
<i>rvtoHideGridLines</i>	If not set (default), invisible borders are shown using GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle ⁽⁶⁴⁰⁾ properties of the linked TRVStyle. It's not recommended to use this option. Use rvShowGridLines option in TRichView.Options ⁽²⁴⁰⁾ instead.
<i>rvtoCellBelowBorders</i>	If not set (default), space below invisible side ⁽⁹⁰⁶⁾ of border has color ⁽⁸⁵⁸⁾ of tables. If set, color of cell ⁽⁹⁰⁰⁾ is used. 

Option	Meaning
<i>rvtoOverlappingCorners</i>	<p>If not set (default), colored borders (CellBorderStyle⁸⁵⁴ =<i>rvtbColor</i>) have three-dimensional appearance. If not set, they are flat.</p> <p>The effect can be noticed for wide borders with some invisible sides⁹⁰⁶.</p> <div style="display: flex; align-items: center; margin-top: 10px;">  <div style="margin-left: 10px;"> <p>Standard corners</p> <p>Overlapping corners</p> </div> </div>

RTF and DocX Export Options

Option	Meaning
<i>rvtoRTFAllowAutofit</i>	<p>If set, tables with default width (BestWidth⁸⁵⁰ =0) are exported to RTF and DocX as fit-by-content tables. Off by default.</p> <p>See also: RichViewTableDefaultRTFAutofit¹⁰⁴⁹</p>
<i>rvtoRTFSaveCellPixelBestWidth</i>	<p>If set, exact sizes of cells are saved to RTF and DocX. Off by default.</p>

The options above are explained in the topic about table RTF and DocX export²⁰³.

7.1.1.31 TRVTableItemInfo.PrintOptions

Specifies options for table printing.

type

```
TRVTablePrintOption =
  (rvtoHalftoneBorders, rvtoRowsSplit,
   rvtoWhiteBackground, rvtoContinue);
TRVTablePrintOptions = set of TRVTablePrintOption;
```

property PrintOptions: TRVTablePrintOptions;

Option	Meaning
<i>rvtoHalftoneBorders</i>	<p>If set (default), table and cells borders will be printed in dithered colors, but corners will be rounded.</p> <p>If cleared, borders will be printed in solid colors (usually black or white), but corners will be strictly defined, like on screen.</p>

Option	Meaning
	Affects to 3d borders (see BorderStyle ⁸⁵² and CellBorderStyle ⁸⁵⁴) and borders with some invisible lines. Forced compromise.
<i>rvtoRowsSplit</i>	Clear this option to forbid printing this table on several pages.
<i>rvtoWhiteBackground</i>	If set, colors of table and cells background will be ignored and printed in white. See also TRVPrint.TransparentBackground ⁸²⁷
<i>rvtoContinue</i>	If set, and the table has heading rows ⁸⁵⁹ , heading rows are displayed only from the second table page

Default value:

[*rvtoHalftoneBorders*,*rvtoRowsSplit*]

See also:

- [HeadingRowCount](#)⁸⁵⁹.

7.1.1.32 TRVTableItemInfo.RowCount

Returns the number of rows in the table.

property `RowCount: Integer`

(introduced in v10)

See also properties:

- [ColCount](#)⁸⁵⁸;
- [Cells](#)⁸⁵⁷;
- [Rows](#)⁸⁶⁴.

7.1.1.33 TRVTableItemInfo.Rows

Provides indexed access to all rows in the table.

property `Rows: TRVTableRows`⁹¹¹;

`Rows` is a list of table rows ([TRVTableRow](#)⁹⁰⁹), each row is a list of cells ([TRVTableCellData](#)⁸⁹⁵). Each row has the same number of cells.

The specific cell can be accessed as `Rows[<Row>][<Col>]` as well as `Cells`⁸⁵⁷[<Row>,<Col>].

Use `Rows` to access properties of rows, for example `Rows[<Row>].VAlign`⁹¹⁰. Otherwise, `Cells`⁸⁵⁷ is more convenient.

See also:

- [Table Cells Overview](#)¹⁹²

7.1.1.34 TRVTableItemInfo.TextColSeparator

Value of this property is used when saving table in text file. This string will be written between cells.

```
property TextColSeparator: TRVUnicodeString1032;
```

(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:

- TextRowSeparator⁸⁶⁵.

See also methods of TRichView:

- SaveText³⁴⁵;
- GetSelText³¹³.

See also:

- Export of Tables²⁰³.

7.1.1.35 TRVTableItemInfo.TextRowSeparator

Value of this property is used when saving table in text file. This string will be written between rows.

```
property TextRowSeparator: TRVUnicodeString1032;
```

(changed in version 18)

Default value:

#13#10 (CR+LF)

See also properties:

- TextColSeparator⁸⁶⁵.

See also methods of TRichView:

- SaveText³⁴⁵;
- GetSelText³¹³.

See also:

- Export of Tables²⁰³.

7.1.1.36 TRVTableItemInfo.VisibleBorders

Allows to hide some sides of table border.

```
property VisibleBorders: TRVBooleanRect961;
```

(introduced in v10)

Direct assignment to this property cannot be undone/redone by user (use `SetTableVisibleBorders` ⁸⁸⁹)

Default value:

True, True, True, True (all sides are visible)

See also properties:

- `BorderWidth` ⁸⁵³;
- `BorderColor` ⁸⁵¹;
- `BorderLightColor` ⁸⁵²;
- `BorderStyle` ⁸⁵².

See also properties of table cell:

- `VisibleBorders` ⁹⁰⁶.

7.1.1.37 TRVTableItemInfo.VOutermostRule

Specifies if the vertical rules should be drawn between the table border and the outermost cells.

property `VOutermostRule`: `Boolean`;

These rules can be visible only if other rules are visible (`VRuleWidth` ⁸⁶⁷ > 0)

Note about rules.

Default value:

False

See also:

- Undo of assignments to table properties;
- Table rules ¹⁹⁴ (with pictures).

See also properties:

- `HOutermostRule` ⁸⁵⁹;
- `VRuleWidth` ⁸⁶⁷;
- `VRuleColor` ⁸⁶⁶.

7.1.1.38 TRVTableItemInfo.VRuleColor

Specifies the color of vertical rules (vertical lines between cells)

property `VRuleColor`: `TRVColor` ⁹⁹⁶;

Vertical rules are visible only if `VRuleWidth` ⁸⁶⁷ > 0

Note about rules.

Default value:

rvclWindowText ¹⁰³⁸

See also:

- Undo of assignments to table properties;
- Table rules ¹⁹⁴ (with pictures).

See also properties:

- HRuleColor⁸⁶⁰;
- VRuleWidth⁸⁶⁷;
- VOutermostRule⁸⁶⁶.

7.1.1.39 TRVTableItemInfo.VRuleWidth

Specifies the width of vertical rules (vertical lines between cells)

```
property VRuleWidth: TRVStyleLength1027;
```

This value is measured in Units⁶⁵⁵ of the linked TRVStyle⁶³⁰ 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 rules¹⁹⁴ (with pictures).

See also properties:

- HRuleWidth⁸⁶⁰;
- VRuleColor⁸⁶⁶;
- VOutermostRule⁸⁶⁶.

7.1.1.40 TRVTableItemInfo's row and column colors

A set of properties defining default colors of cells in specific rows and columns.

```
property HeadingRowColor: TRVColor996 ;  
property LastRowColor: TRVColor996 ;  
property OddRowsColor: TRVColor996 ;  
property EvenRowsColor: TRVColor996 ;  
property FirstColumnColor: TRVColor996 ;  
property LastColumnColor: TRVColor996 ;  
property OddColumnsColor: TRVColor996 ;  
property EvenColumnsColor: TRVColor996 ;
```

(introduced in version 17)

By default, all cells are transparent, and you can see the table background through cells (see Color⁸⁵⁸ 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.Color⁹⁰⁰ property.

Property	Meaning
HeadingRowColor	Color of heading rows ⁽⁸⁵⁹⁾ . If this table has no heading rows, but HeadingRowColor <> <i>c/None</i> , 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**<>*c/None*, column bands are started from the second column, otherwise from the first column. Each band has **ColBandSize** ⁽⁸⁵⁸⁾ columns.

Row bands are started from the first column after the table heading rows. If **HeadingRowCount** ⁽⁸⁵⁹⁾ >0, it defines the number of heading rows. Otherwise, if **HeadingRowColor**<>*c/None*, one heading row is assumed. Otherwise, row bands start from the first row. Each band has **RowBandSize** ⁽⁸⁵⁸⁾ 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 *c/None* are taken into account.

Default values

rvc/None ⁽¹⁰³⁸⁾

See also

- **Color** ⁽⁸⁵⁸⁾
- **CellOverrideColor** ⁽⁸⁵⁶⁾ ;
- **HeadingRowCount** ⁽⁸⁵⁹⁾
- **ColBandSize**, **RowBandSize** ⁽⁸⁵⁸⁾
- **Cell.Color** ⁽⁹⁰⁰⁾
- **Cell.Opacity** ⁽⁹⁰²⁾

7.1.2 Methods

In TRVTableItemInfo

AssignProperties ⁽⁸⁷⁰⁾
CanMergeCells ⁽⁸⁷⁰⁾
CanMergeSelectedCells ⁽⁸⁷¹⁾
Changed ⁽⁸⁷¹⁾
Create ⁽⁸⁷²⁾

CreateEx⁸⁷²
DeleteCols⁸⁷³
DeleteEmptyCols⁸⁷³
DeleteEmptyRows⁸⁷³
DeleteRows⁸⁷⁴
DeleteSelectedRows⁸⁷⁵
DeleteSelectedCols⁸⁷⁴
Deselect⁸⁷⁵
Destroy⁸⁷⁶
EditCell⁸⁷⁶
GetCellAt⁸⁷⁶
GetEditedCell⁸⁷⁷
GetNormalizedSelectionBounds⁸⁷⁷
GetSelectionBounds⁸⁷⁸
InsertCols⁸⁷⁸
InsertColsLeft⁸⁷⁹
InsertColsRight⁸⁷⁹
InsertRows⁸⁸⁰
InsertRowsAbove⁸⁸⁰
InsertRowsBelow⁸⁸⁰
IsCellSelected⁸⁸¹
LoadFromStream⁸⁸¹
MergeCells⁸⁸¹
MergeSelectedCells⁸⁸²
MoveRows⁸⁸²
SaveRowsToStream⁸⁸³
SaveToStream⁸⁸³
Select⁸⁸⁴
SelectCols⁸⁸⁴
SelectRows⁸⁸⁴
SetCellBackgroundImage⁸⁸⁵
SetCellBackgroundImageFileName⁸⁸⁵
SetCellBackgroundStyle⁸⁸⁵
SetCellBestHeight⁸⁸⁶
SetCellBestWidth⁸⁸⁶
SetCellBorderColor⁸⁸⁶
SetCellBorderLightColor⁸⁸⁶
SetCellColor⁸⁸⁷
SetCellHint⁸⁸⁷
SetCellOpacity⁸⁸⁷
SetCellOptions⁸⁸⁷
SetCellRotation⁸⁸⁸
SetCellTag⁸⁸⁸
SetCellVAlign⁸⁸⁸
SetCellVisibleBorders⁸⁸⁹
SetRowKeepTogether⁸⁸⁹
SetRowPageBreakBefore⁸⁸⁹
SetRowVAlign⁸⁸⁹

SetTableVisibleBorders⁽⁸⁸⁹⁾
 SplitSelectedCellsHorizontally⁽⁸⁹⁰⁾
 SplitSelectedCellsVertically⁽⁸⁹⁰⁾
 UnmergeCells⁽⁸⁹⁰⁾
 UnmergeSelectedCells⁽⁸⁹¹⁾

7.1.2.1 TRVTableItemInfo.AssignProperties

Assigns all table properties from the **Source**.

```
procedure AssignProperties(Source: TRVTableItemInfo(846));
```

(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⁽⁸⁵⁷⁾ [**TopRow, TopCol**] with ColSpan⁽⁹⁰⁰⁾ = **ColSpan** and RowSpan⁽⁹⁰⁵⁾ = **RowSpan**.

Merging can be done, if

1. Cells⁽⁸⁵⁷⁾ [**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⁽⁸⁵⁷⁾ [**TopRow, TopCol**].ColSpan⁽⁹⁰⁰⁾ < **ColSpan** or Cells⁽⁸⁵⁷⁾ [**TopRow, TopCol**].RowSpan⁽⁹⁰⁵⁾ < **RowSpan**.
4. If **AllowMergeRC=True**, this function returns *False*, if one or more rows or columns will contain only *nils* after merging. It's not recommended to have such rows/columns, so you can either forbid such operation or (better) delete them with DeleteEmptyRows⁽⁸⁷³⁾ and DeleteEmptyCols⁽⁸⁷³⁾.

See also:

- CanMergeSelectedCells⁽⁸⁷¹⁾;
- MergeCells⁽⁸⁸¹⁾;
- MergeSelectedCells⁽⁸⁸²⁾.

See also properties:

- ColCount⁽⁸⁵⁸⁾;

- RowCount⁽⁸⁶⁴⁾.

7.1.2.3 TRVTableItemInfo.CanMergeSelectedCells

Returns: is cell-merging of the selected cells possible?

```
function CanMergeSelectedCells(AllowMergeRC: Boolean): Boolean;
```

The same as CanMergeCells⁽⁸⁷⁰⁾, 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 *nils* after merging. It's not recommended to have such rows/columns, so you can either forbid such operation or (better) delete them with DeleteEmptyRows⁽⁸⁷³⁾ and DeleteEmptyCols⁽⁸⁷³⁾.

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⁽⁸⁷⁰⁾;
- MergeSelectedCells⁽⁸⁸²⁾;
- MergeCells⁽⁸⁸¹⁾.

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⁽⁴⁹⁵⁾/EndItemModify⁽⁵⁰³⁾ (if the table was inserted in TRichViewEdit⁽⁴⁶¹⁾) or Format⁽²⁸⁸⁾ (if the table was inserted in TRichView⁽²¹⁰⁾).

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(296)(ItemNo));  
table.Cells[0,0].Clear(279);  
table.Cells[0,0].AddNL(908)('One more line',0,0);  
table.Changed;  
MyRichView.Format(288);
```

Note: since version 1.7, it's not required to call table.Changed before Format⁽²⁸⁸⁾. But it is still useful when using BeginItemModify⁽⁴⁹⁵⁾/EndItemModify⁽⁵⁰³⁾.

7.1.2.5 TRVTableItemInfo.Create

Creates a new TRVTableItemInfo object

constructor Create(RVData: TPersistent); **override**;

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, 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⁽⁸⁷²⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

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⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, 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⁽⁸⁶⁴⁾>0.

Example

```
var table:TRVTableItemInfo;
...
table := TRVTableItemInfo.CreateEx(5, 5, MyRichViewEdit.RVData(247));
table.Color(858) := clYellow;
table.Cells(857)[0,0].AddNL(908)('Hello',0,0);
MyRichViewEdit.InsertItem(522)('5x5 table', table);
```

See also methods:

- Create⁽⁸⁷²⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

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..ColCount⁸⁵⁸-1.

Count – number of columns to delete.

If **DecreaseWidth** = *True*, BestWidth⁸⁹⁸s 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 operations¹⁹⁹;
- Undo note.

See also methods:

- DeleteSelectedCols⁸⁷⁴;
- DeleteRows⁸⁷⁴;
- DeleteEmptyCols⁸⁷³.

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 DeleteCols⁸⁷³ with *DecreaseWidth* = *False*.

The method keeps selection in the table.

See also:

- Table operations¹⁹⁹;
- Undo note.

See also methods:

- DeleteEmptyRows⁸⁷³;
- DeleteSelectedCols⁸⁷⁴;
- DeleteCols⁸⁷³.

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`⁽⁸⁷⁴⁾ with `DecreaseHeight = False`.

The method keeps selection in the table.

See also:

- Table operations⁽¹⁹⁹⁾;
- Undo note.

See also:

- `DeleteEmptyCols`⁽⁸⁷³⁾;
- `DeleteSelectedRows`⁽⁸⁷⁵⁾;
- `DeleteRows`⁽⁸⁷⁴⁾.

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`⁽⁸⁶⁴⁾-1.

Count – number of rows to delete.

if **DecreaseHeight** = *True*, `BestHeight`⁽⁸⁹⁸⁾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⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- `DeleteSelectedRows`⁽⁸⁷⁵⁾;
- `DeleteCols`⁽⁸⁷³⁾;
- `DeleteEmptyRows`⁽⁸⁷³⁾.

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 ⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- DeleteSelectedRows ⁽⁸⁷⁵⁾;
- DeleteCols ⁽⁸⁷³⁾;
- DeleteEmptyCols ⁽⁸⁷³⁾.

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 ⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- DeleteSelectedCols ⁽⁸⁷⁴⁾;
- DeleteCols ⁽⁸⁷³⁾;
- DeleteEmptyCols ⁽⁸⁷³⁾.

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 ⁽¹⁹⁵⁾.

See also methods:

- Select ⁽⁸⁸⁴⁾;
- SelectRows ⁽⁸⁸⁴⁾;
- SelectCols ⁽⁸⁸⁴⁾.

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⁽⁸⁶⁴⁾-1, **Col** must be in range 0..ColCount⁽⁸⁵⁸⁾-1.

Cells⁽⁸⁷⁶⁾[**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⁽⁸⁷⁶⁾[Row, Col].Edit⁽⁹⁵⁹⁾.

This method must be called when the document is formatted.

See also:

- Overview of cell editing⁽¹⁹⁷⁾.

See also methods:

- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾;
- GetEditedCell⁽⁸⁷⁷⁾.

7.1.2.16 TRVTableItemInfo.GetCellAt

Returns cell at the specified coordinates.

function GetCellAt(X, Y: TRVCoord⁽⁹⁹⁸⁾; **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⁽²⁹⁸⁾.

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(461);
```

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⁽¹⁹⁵⁾;
- Overview of cell editing⁽¹⁹⁷⁾.

See also methods:

- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾;
- EditCell⁽⁸⁷⁶⁾.

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(857)[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⁽⁸⁸¹⁾.

Some operations are possible only on rectangular selection (for example, cell merging). It can be checked using CanMergeSelectedCells⁽⁸⁷¹⁾.

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⁽¹⁹⁵⁾

See also methods:

- GetSelectionBounds⁽⁸⁷⁸⁾;
- GetEditedCell⁽⁸⁷⁷⁾.

7.1.2.19 TRVTableItemInfo.GetSelectionBounds

Returns selection in table.

```
function GetSelectionBounds(out StartRow, StartCol,
    RowOffs, ColOffs: Integer): Boolean;
```

Output parameters:

Cells⁽⁸⁵⁷⁾[**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⁽⁸⁷¹⁾.

This method returns selection as it is stored internally. For practical use, GetNormalizedSelectionBounds⁽⁸⁷⁷⁾ 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⁽¹⁹⁵⁾.

See also methods:

- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾;
- Select⁽⁸⁸⁴⁾;
- Deselect⁽⁸⁷⁵⁾.

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⁽⁸⁵⁸⁾ 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 ⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- InsertRows ⁽⁸⁸⁰⁾;
- InsertColsLeft ⁽⁸⁷⁹⁾, InsertColsRight ⁽⁸⁷⁹⁾;
- DeleteCols ⁽⁸⁷³⁾.

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 ⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- InsertColsRight ⁽⁸⁷⁹⁾;
- InsertCols ⁽⁸⁷⁸⁾;
- InsertRowsAbove ⁽⁸⁸⁰⁾, InsertRowsBelow ⁽⁸⁸⁰⁾;
- DeleteSelectedCols ⁽⁸⁷⁴⁾.

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 ⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- InsertColsLeft ⁽⁸⁷⁹⁾;
- InsertCols ⁽⁸⁷⁸⁾;
- InsertRowsAbove ⁽⁸⁸⁰⁾, InsertRowsBelow ⁽⁸⁸⁰⁾;
- DeleteSelectedCols ⁽⁸⁷⁴⁾.

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 RowCount⁸⁶⁴ 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 operations¹⁹⁹;
- Undo note.

See also methods:

- InsertCols⁸⁷⁸;
- InsertRowsAbove⁸⁸⁰, InsertRowsBelow⁸⁸⁰;
- DeleteRows⁸⁷⁴.

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 operations¹⁹⁹;
- Undo note.

See also methods:

- InsertRowsBelow⁸⁸⁰;
- InsertRows⁸⁸⁰;
- InsertColsLeft⁸⁷⁹, InsertColsRight⁸⁷⁹;
- DeleteSelectedRows⁸⁷⁵.

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 operations¹⁹⁹;
- Undo note.

See also methods:

- InsertRowsAbove⁸⁸⁰;

- `InsertRows`⁽⁸⁸⁰⁾.
- `InsertColsLeft`⁽⁸⁷⁹⁾, `InsertColsRight`⁽⁸⁷⁹⁾;
- `DeleteSelectedRows`⁽⁸⁷⁵⁾.

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(864) - 1 do
  for c := 0 to table.ColCount(858) - 1 do
    if (table.Cells(857)[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⁽¹⁹⁵⁾.

See also methods:

- `GetNormalizedSelectionBounds`⁽⁸⁷⁷⁾.

7.1.2.27 TRVTableItemInfo.LoadFromStream

Loads table from **Stream**.

```
procedure LoadFromStream(Stream: TStream);
```

Table must be saved using `SaveToStream`⁽⁸⁸³⁾ or `SaveRowsToStream`⁽⁸⁸³⁾ methods. Table can be loaded correctly only if collections of text, paragraph and list styles in `TRVStyle`⁽⁶³⁰⁾ 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⁽¹²⁴⁾ 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(522)(' ', 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⁽⁸⁵⁷⁾ [**TopRow**, **TopCol**] having ColSpan⁽⁹⁰⁰⁾ = **ColSpan** and RowSpan⁽⁹⁰⁵⁾ = **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⁽⁸⁷⁰⁾.

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⁽⁸⁷³⁾ and DeleteEmptyCols⁽⁸⁷³⁾ (the latter is recommended).

See also:

- Table operations⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- CanMergeCells⁽⁸⁷⁰⁾;
- MergeSelectedCells⁽⁸⁸²⁾;
- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾;
- UnmergeCells⁽⁸⁹⁰⁾.

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⁽⁸⁷¹⁾.

AllowMergeRC = *False* forbids to perform merging, if it will cause appearing rows or columns containing only *nils*. It's not recommended to have such rows/columns, so you can either forbid such operation or delete them with DeleteEmptyRows⁽⁸⁷³⁾ and DeleteEmptyCols⁽⁸⁷³⁾ (the latter is recommended).

This method must be called when the document is formatted.

See also:

- Table operations⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- CanMergeSelectedCells⁽⁸⁷¹⁾;
- MergeCells⁽⁸⁸¹⁾;
- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾;
- UnmergeSelectedCells⁽⁸⁹¹⁾.

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 .. RowCount⁸⁶⁴ - ARowCount)

ARowCount specifies how many rows will be moved (a positive value)

AToRow – where to move (in range 0 .. RowCount⁸⁶⁴ - 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 operations¹⁹⁹;
- 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 SaveToStream⁸⁸³.

The result will be correct even if there are cells that overlap these rows because of RowSpan⁹⁰⁵ > 1.

See also method:

- LoadFromStream⁸⁸¹.

See also:

- Example how to divide table in two parts²⁰⁸.

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 streams¹²⁴ with main documents.

See also method:

- LoadFromStream⁸⁸¹;
- SaveRowsToStream⁸⁸³.

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⁽⁸⁷⁸⁾.

Selection includes all cells intersected with imaginary rectangle that fully includes Cells⁽⁸⁵⁷⁾ [**StartRow**, **StartCol**] and Cells⁽⁸⁵⁷⁾ [**StartRow+RowOffs**, **StartCol+ColOffs**].

You cannot remove selection with this method, because Select(StartRow, StartCol, 0,0) selects one cell, use Deselect⁽⁸⁷⁵⁾.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables⁽¹⁹⁵⁾

See also methods:

- SelectRows⁽⁸⁸⁴⁾, SelectCols⁽⁸⁸⁴⁾;
- GetSelectionBounds⁽⁸⁷⁸⁾;
- Deselect⁽⁸⁷⁵⁾.

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⁽⁸⁵⁸⁾-1.

Count – number of columns to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables⁽¹⁹⁵⁾

See also methods:

- SelectRows⁽⁸⁸⁴⁾,
- Select⁽⁸⁸⁴⁾.

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⁽⁸⁶⁴⁾-1.

Count – number of rows to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables ⁽¹⁹⁵⁾

See also methods:

- SelectCols ⁽⁸⁸⁴⁾;
- Select ⁽⁸⁸⁴⁾.

7.1.2.36 TRVTableItemInfo.SetCellBackgroundImage

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundImage ⁽⁸⁹⁶⁾, as an editing operation.

```
procedure SetCellBackgroundImage(Value: TRVGraphic (970); Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundImage ⁽⁸⁹⁶⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

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 ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.37 TRVTableItemInfo.SetCellBackgroundImageFileName

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundImageFileName ⁽⁸⁹⁷⁾, as an editing operation.

```
procedure SetCellBackgroundImageFileName(  
  const Value: TRVUnicodeString (1032); Row, Col: Integer);
```

(changed in version 18)

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundImageFileName ⁽⁸⁹⁷⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.38 TRVTableItemInfo.SetCellBackgroundStyle

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundStyle ⁽⁸⁹⁷⁾, as an editing operation.

```
procedure SetCellBackgroundStyle(Value: TRVItemBackgroundStyle (1013);  
  Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].BackgroundStyle ⁽⁸⁹⁷⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.39 TRVTableItemInfo.SetCellBestHeight

Assigns **Value** to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BestHeight ⁽⁸⁹⁸⁾, as an editing operation.

```
procedure SetCellBestHeight(Value: TRVStyleLength(1027); Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BestHeight ⁽⁸⁹⁸⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.40 TRVTableItemInfo.SetCellBestWidth

Assigns **Value** to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BestWidth ⁽⁸⁹⁸⁾, as an editing operation.

```
procedure SetCellBestWidth(Value: TRVHTMLLength(1012); Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BestWidth ⁽⁸⁹⁸⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.41 TRVTableItemInfo.SetCellBorderColor

Assigns **Value** to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BorderColor ⁽⁸⁹⁹⁾, as an editing operation.

```
procedure SetCellBorderColor(Value: TRVColor(996); Row, Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BorderColor ⁽⁸⁹⁹⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.42 TRVTableItemInfo.SetCellBorderLightColor

Assigns **Value** to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BorderLightColor ⁽⁸⁹⁹⁾, as an editing operation.

```
procedure SetCellBorderLightColor(Value: TRVColor(996); Row, Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾[**Row, Col**].BorderLightColor ⁽⁸⁹⁹⁾ property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.43 TRVTableItemInfo.SetCellColor

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].Color ⁽⁹⁰⁰⁾, as an editing operation.

```
procedure SetCellColor(Value: TRVColor (996); Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].Color ⁽⁹⁰⁰⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.44 TRVTableItemInfo.SetCellHint

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].Hint ⁽⁹⁰¹⁾, as an editing operation.

```
procedure SetCellHint(const Value: TRVUnicodeString (1032); Row, Col: Integer);
```

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].Hint ⁽⁹⁰¹⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.45 TRVTableItemInfo.SetCellOpacity

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].Opacity ⁽⁹⁰²⁾, as an editing operation.

```
procedure SetCellOpacity(Value: TRVOpacity (1015); Row, Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells ⁽⁸⁵⁷⁾ [Row, Col].Opacity ⁽⁹⁰²⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion ⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example ⁽²⁰²⁾.

See also:

- Undo/redo in tables ⁽²⁰²⁾;
- Undo/redo in TRichViewEdit ⁽¹¹⁵⁾.

7.1.2.46 TRVTableItemInfo.SetCellOptions

Assigns **Value** to Cells ⁽⁸⁵⁷⁾ [Row, Col].Options ⁽⁹⁰²⁾, as an editing operation.

```
procedure SetCellOptions(Value: TRVTableCellOptions (902); Row, Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells⁽⁸⁵⁷⁾ [Row, Col].Options⁽⁹⁰²⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.47 TRVTableItemInfo.SetCellRotation

Assigns **Value** to Cells⁽⁸⁵⁷⁾ [Row, Col].Rotation⁽⁹⁰⁴⁾, as an editing operation.

```
procedure SetCellRotation(Value: TRVDocRotation(999); Row, Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells⁽⁸⁵⁷⁾ [Row, Col].Rotation⁽⁹⁰⁴⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.48 TRVTableItemInfo.SetCellTag

Assigns **Value** to Cells⁽⁸⁵⁷⁾ [Row, Col].Tag⁽⁹⁰⁶⁾, as an editing operation.

```
procedure SetCellVAlign(Value: Tag(906); Row, Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells⁽⁸⁵⁷⁾ [Row, Col].Tag⁽⁹⁰⁶⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾
- Tags⁽⁹¹⁾

7.1.2.49 TRVTableItemInfo.SetCellVAlign

Assigns **Value** to Cells⁽⁸⁵⁷⁾ [Row, Col].VAlign⁽⁹⁰⁶⁾, as an editing operation.

```
procedure SetCellVAlign(Value: TRVCellVAlign(996); Row, Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells⁽⁸⁵⁷⁾ [Row, Col].VAlign⁽⁹⁰⁶⁾ property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;

- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.50 TRVTableItemInfo.SetCellVisibleBorders

Assigns values to Cells⁽⁸⁵⁷⁾ [Row, Col].VisibleBorders⁽⁹⁰⁶⁾, as an editing operation.

```
procedure SetCellVisibleBorders(Left, Top, Right, Bottom: Boolean;  
Row, Col: Integer);
```

(introduced in version 1.6)

Unlike direct assignment to Cells⁽⁸⁵⁷⁾ [Row, Col].VisibleBorders⁽⁹⁰⁶⁾ subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.51 TRVTableItemInfo.SetRow*

Assigns **Value** to Rows⁽⁸⁶⁴⁾ [Row].VAlign⁽⁹¹⁰⁾ (or PageBreakBefore⁽⁹¹⁰⁾, or KeepTogether⁽⁹¹⁰⁾), as an editing operation.

```
procedure SetRowVAlign(Value: TRVCellVAlign(996); 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⁽⁸⁶⁴⁾ [Row] property, these methods can be undone/redone, if they were called for a table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use these methods in conjunction with the methods for updating an editor window, see the example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.52 TRVTableItemInfo.SetTableVisibleBorders

Assigns values to VisibleBorders⁽⁸⁶⁵⁾, as an editing operation.

```
procedure SetTableVisibleBorders(Left, Top, Right, Bottom: Boolean);
```

(introduced in version 10)

Unlike direct assignment to VisibleBorders⁽⁸⁶⁵⁾ subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion⁽⁵²²⁾).

Use this method in conjunction with methods for updating editor window, see example⁽²⁰²⁾.

See also:

- Undo/redo in tables⁽²⁰²⁾;
- Undo/redo in TRichViewEdit⁽¹¹⁵⁾.

7.1.2.53 TRVTableItemInfo.SplitSelectedCellsHorizontally

Splits each selected cell into **RowCount** cells (inserting horizontal splitters).

```
procedure SplitSelectedCellsHorizontally(RowCount: Integer);
```

This method splits cells calling `UnmergeCells`⁽⁸⁹⁰⁾, `InsertRows`⁽⁸⁸⁰⁾ and `MergeCells`⁽⁸⁸¹⁾. This method can change number of rows.

It's recommended to call this method only when selection has a rectangular shape, see `CanMergeSelectedCells`⁽⁸⁷¹⁾.

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⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- `SplitSelectedCellsVertically`⁽⁸⁹⁰⁾;
- `UnmergeSelectedCells`⁽⁸⁹¹⁾;
- `GetNormalizedSelectionBounds`⁽⁸⁷⁷⁾.

7.1.2.54 TRVTableItemInfo.SplitSelectedCellsVertically

Splits each selected cell into **ColCount** cells (inserting vertical splitters).

```
procedure SplitSelectedCellsVertically(ColCount: Integer);
```

This method splits cells calling `UnmergeCells`⁽⁸⁹⁰⁾, `InsertCols`⁽⁸⁷⁸⁾ and `MergeCells`⁽⁸⁸¹⁾. This method can change number of columns.

It's recommended to call this method only when selection has a rectangular shape, see `CanMergeSelectedCells`⁽⁸⁷¹⁾.

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⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- `SplitSelectedCellsVertically`⁽⁸⁹⁰⁾;
- `UnmergeSelectedCells`⁽⁸⁹¹⁾;
- `GetNormalizedSelectionBounds`⁽⁸⁷⁷⁾.

7.1.2.55 TRVTableItemInfo.UnmergeCells

Unmerges cells.

```
procedure UnmergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer;  
UnmergeRows, UnmergeCols: Boolean);
```

Cells can be merged using the methods `MergeCells`⁽⁸⁸¹⁾ or `MergeSelectedCells`⁽⁸⁸²⁾.

This method unmerges all cells in range from Cells⁽⁸⁵⁷⁾ [**TopRow, LeftCol**] to Cells⁽⁸⁵⁷⁾ [**TopRow+RowSpan, LeftCol+ColSpan**].

If **UnmergeRows**=*True*, it unmerges each cell into RowSpan⁽⁹⁰⁵⁾ rows.

If **UnmergeCols**=*True*, it unmerges each cells into ColSpan⁽⁹⁰⁰⁾ columns.

UnmergeRows and *UnmergeCells* can both be *True*.

See also:

- Table operations⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- UnmergeSelectedCells⁽⁸⁹¹⁾;
- MergeCells⁽⁸⁸¹⁾.

7.1.2.56 TRVTableItemInfo.UnmergeSelectedCells

Unmerges the selected cells.

```
procedure UnmergeSelectedCells(UnmergeRows, UnmergeCols: Boolean);
```

Cells can be merged using the methods MergeCells⁽⁸⁸¹⁾ or MergeSelectedCells⁽⁸⁸²⁾.

If **UnmergeRows**=*True*, it unmerges each selected cell into RowSpan⁽⁹⁰⁵⁾ rows.

If **UnmergeCols**=*True*, it unmerges each selected cells into ColSpan⁽⁹⁰⁰⁾ columns.

UnmergeRows and **UnmergeCells** can both be *True*.

This method must be called when the document is formatted.

See also:

- Table operations⁽¹⁹⁹⁾;
- Undo note.

See also methods:

- UnmergeCells⁽⁸⁹⁰⁾;
- MergeSelectedCells⁽⁸⁸²⁾;
- GetNormalizedSelectionBounds⁽⁸⁷⁷⁾.

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⁽⁸⁹²⁾

OnDrawBackground⁽⁸⁹³⁾

OnDrawBorder⁽⁸⁹³⁾

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**⁽⁸⁵⁷⁾ [**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⁽¹²⁴⁾ 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(269) or InsertItem(522) methods, or is loaded from a file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo(844); var Text: String;
  RVData: TCustomRVData);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
    TRVTableItemInfo(846)(Item).OnCellEditing := DoCellEditing;
end;
```

See also:

- TParaInfo.Options⁽⁷²³⁾ (*rvpaoReadOnly, rvpaoStyleProtect, rvpaoDoNotWantReturns* – protection options for paragraphs);
- TFontInfo.Protection⁽⁷⁰⁵⁾ (protection options for text);

- TRVExtralItemProperty⁽¹⁰⁰⁰⁾ (*rvepDeleteProtect*, *rvepResizable* – protection option for nontext items).

7.1.3.2 TRVTableItemInfo.OnDrawBackground

Allows custom drawing of table backgrounds

type

```
TRVTableDrawBackEvent = procedure (Sender: TRVTableItemInfo(846);
  Canvas: TCanvas; Left, Top, Right, Bottom: TRVCoord(998);
  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⁽⁸⁹³⁾

7.1.3.3 TRVTableItemInfo.OnDrawBorder

Allows custom drawing of table borders

type

```
TRVTableDrawBorderEvent = procedure (Sender: TRVTableItemInfo;
  Canvas: TCanvas; Left, Top, Right, Bottom: TRVCoord(998);
  Width: TRVStyleLength(1027);
  LightColor, Color, BackgroundColor: TRVColor(996);
  Style: TRVTableBorderStyle(1029); Printing: Boolean;
  VisibleBorders: TRVBooleanRect(961); 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`⁽²⁵³⁾.Units⁽⁶⁵⁵⁾.

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⁽⁸⁶⁵⁾ property for the table, or VisibleBorders⁽⁹⁰⁶⁾ 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(855)=1, CellVSpacing(857)=0, CellHSpacing(856)=0.
  This example ignores VisibleBorders.
}
```

```
procedure TMyForm.DoDrawBorder(Sender: TRVTableItemInfo;
  Canvas: TCanvas; Left, Top, Right, Bottom: TRVCoord(998);
  Width: TRVStyleLength(1027);
  LightColor, Color, BackgroundColor: TRVColor(996);
  Style: TRVTableBorderStyle(1029);
  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⁽¹²⁴⁾ files, so you need to reassign them when tables are loaded. The best place to assign this event is TRichView.OnItemAction event.

```
// MyRichViewEdit.OnItemAction(380). Assigns DoDrawBorder to
```

```
// OnDrawBorder events of all tables inserted in MyRichViewEdit.
// (this assignment occurs when table is inserted using
// AddItem269 or InsertItem522 methods, or is loaded from RVF or RTF file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView210;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo844;
  var Text: TRVUnicodeString1032; RVDData: TCustomRVData957);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
    TRVTableItemInfo846(Item).OnDrawBorder := DoDrawBorder;
end;
```

See also

- OnDrawBackground⁸⁹³

7.1.4 Classes of properties

Classes of TRVTableItemInfo⁸⁴⁶ Properties

- TRVTableRows⁹¹¹ – a list of table rows (TRVTableRow⁹⁰⁹). This is a type of Rows⁸⁶⁴ property.
- TRVTableRow⁹⁰⁹ – a list of table cells (TRVTableCellData⁸⁹⁵). Accessible as Rows[r]⁹¹².
- TRVTableCellData⁸⁹⁵ – a table cell. Accessible as Cells[r, c]⁸⁵⁷

7.1.4.1 TRVTableCellData

TRVTableCellData represents one cell in a table (see TRVTableItemInfo⁸⁴⁶). Cells can be accessed using Cells⁸⁵⁷ property of table.

Unit RVTable.

Syntax

```
TRVTableCellData = class(TRVItemFormattedData957)
```

Hierarchy

TObject

TPersistent

TCustomRVData⁹⁵⁷

TCustomRVFormattedData

TRVItemFormattedData

Methods and Properties

TRVTableCellData has many properties and methods corresponding to properties and methods⁹⁰⁸ of TRichView²¹⁰.

TRVTableCellData and class of object TRichView.RVData²⁴⁷ are derived from the same base class.

In addition, this class has properties affecting the cell view and layout:

- Color⁹⁰⁰ – background color of cell;
- BackgroundImage⁸⁹⁶, BackgroundStyle⁸⁹⁷ – background image and its position;
- BackgroundImageFileName⁸⁹⁷ – can be used when exporting to HTML;
- BorderColor⁸⁹⁹, BorderLightColor⁸⁹⁹ – colors of border;

- `BestWidth`⁽⁸⁹⁸⁾, `BestHeight`⁽⁸⁹⁸⁾, `IgnoreContentHeight`⁽⁹⁰¹⁾ – preferred size of cell,
- `VAlign`⁽⁹⁰⁶⁾ – vertical alignment of contents;
- `Rotation`⁽⁹⁰⁴⁾ – rotation of contents;
- `VisibleBorders`⁽⁹⁰⁶⁾ – defines visible sides of cell border.
- `Tag`⁽⁹⁰⁶⁾ – a string value (not used by the component itself).

These properties can be saved with cell in RVF⁽¹²⁴⁾ file or stream.

Also, this class has read-only properties:

- `ColSpan`⁽⁹⁰⁰⁾, `RowSpan`⁽⁹⁰⁵⁾.

Important methods:

- `GetRVData`⁽⁹⁰⁷⁾.

See also

Table Cells Overview⁽¹⁹²⁾

7.1.4.1.1 Properties

Inherited from TRVItemFormattedData⁽⁹⁵⁷⁾

Read here⁽⁹⁰⁸⁾

In TRVTableCellData

- `BackgroundImage`⁽⁸⁹⁶⁾
- `BackgroundImageFileName`⁽⁸⁹⁷⁾
- `BackgroundStyle`⁽⁸⁹⁷⁾
- `BestHeight`⁽⁸⁹⁸⁾
- `BestWidth`⁽⁸⁹⁸⁾
- `BorderColor`⁽⁸⁹⁹⁾
- `BorderLightColor`⁽⁸⁹⁹⁾
- `Color`⁽⁹⁰⁰⁾
- ▶ `ColSpan`⁽⁹⁰⁰⁾
- `Hint`⁽⁹⁰¹⁾
- `IgnoreContentHeight`⁽⁹⁰¹⁾
- `Opacity`⁽⁹⁰²⁾
- `Rotation`⁽⁹⁰⁴⁾
- ▶ `RowSpan`⁽⁹⁰⁵⁾
- `Tag`⁽⁹⁰⁶⁾
- `VAlign`⁽⁹⁰⁶⁾
- `VisibleBorders`⁽⁹⁰⁶⁾

7.1.4.1.1.1 TRVTableCellData.BackgroundImage

Background image for the table cell

```
property BackgroundImage: TRVGraphic(970);
```

(introduced in v1.8)

Assignment to this property copies the image, so you still need to free the source image yourself.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see `RichViewMaxPictureCount`⁽¹⁰⁴⁶⁾).

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`⁽⁸⁹⁷⁾;
- `BackgroundStyle`⁽⁸⁹⁷⁾;
- `Color`⁽⁹⁰⁰⁾.

See also properties of table:

- `BackgroundImage`⁽⁸⁴⁹⁾;
- `BackgroundStyle`⁽⁸⁵⁰⁾.

See also properties of TRichView:

- `BackgroundBitmap`⁽²²²⁾;
- `BackgroundStyle`⁽²²³⁾.

7.1.4.1.1.2 TRVTableCellData.BackgroundImageFileName

This is a string for storing file name associated with `BackgroundImage`⁽⁸⁹⁶⁾

```
property BackgroundImageFileName: TRVUnicodeString(1032);
```

(introduced in v1.9; changed in version 18)

If `rvhtmlsioUseItemImageFileNames` in `HTMLSaveProperties`⁽²³⁷⁾.`ImageOptions`⁽⁴³⁴⁾, 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⁽¹⁰⁰⁵⁾.

See also properties of table:

- `BackgroundImageFileName`⁽⁸⁵⁰⁾.

7.1.4.1.1.3 TRVTableCellData.BackgroundStyle

Defines the position of `BackgroundImage`⁽⁸⁹⁶⁾.

```
property BackgroundStyle: TRVItemBackgroundStyle(1013);
```

(introduced in v1.8)

See also:

- `BackgroundImage`⁽⁸⁹⁶⁾;
- `Color`⁽⁹⁰⁰⁾.

See also properties of table:

- BackgroundStyle⁽⁸⁵²⁾.
- BackgroundImage⁽⁸⁴⁹⁾.

See also properties of TRichView:

- BackgroundStyle⁽²²³⁾.
- BackgroundBitmap⁽²²²⁾.

7.1.4.1.1.4 TRVTableCellData.BestHeight

Height for the cell.

property BestHeight: TRVStyleLength⁽¹⁰²⁷⁾;

This value is measured in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ component.

This property corresponds to <td height> in HTML, but there is a difference. **BestHeight** does not include CellVPadding⁽⁸⁵⁷⁾, while <td height> includes vertical cell padding.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBestHeight⁽⁸⁸⁶⁾.

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⁽⁹⁰¹⁾ = *True* for this cell, or if *rvtIgnoreContentHeight* is included in table.Options⁽⁸⁶¹⁾. 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⁽⁸⁹⁸⁾;
- IgnoreContentHeight⁽⁹⁰¹⁾.

See also properties of table:

- Options⁽⁸⁶¹⁾ (*rvtIgnoreContentHeight* option).

See also:

- Table Resizing⁽²⁰⁵⁾.

7.1.4.1.1.5 TRVTableCellData.BestWidth

Preferred width for the cell

property BestWidth: TRVHTMLLength⁽¹⁰¹²⁾;

This is a preferred width for the cell, measured either in Units⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾ 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⁽⁸⁸⁶⁾.

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⁽⁸⁵⁵⁾*2 + paragraph indents (FirstIndent⁽⁷²²⁾, LeftIndent⁽⁷²²⁾, RightIndent⁽⁷²⁵⁾) (if *rvtIgnoreContentWidth* is not included in table.Options⁽⁸⁶¹⁾).

The cell can be narrower than specified in this property, if the table is not wide enough.

Default value:

0

See also properties:

- BestHeight⁽⁸⁹⁸⁾.

See also properties of table:

- BestWidth⁽⁸⁵⁰⁾;
- Options⁽⁸⁶¹⁾ (*rvtIgnoreContentWidth* option).

See also:

- Table Resizing⁽²⁰⁵⁾.

7.1.4.1.1.6 TRVTableCellData.BorderColor

Color of border for this cell.

property BorderColor: TRVColor⁽⁹⁹⁶⁾

(introduced in version 1.6)

If this value of this property is *rvcNone*⁽¹⁰³⁸⁾, the cell uses table.CellBorderColor⁽⁸⁵³⁾.

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⁽⁸⁸⁶⁾.

Default value:

rvcNone⁽¹⁰³⁸⁾

See also properties:

- BorderLightColor⁽⁸⁹⁹⁾;
- Color⁽⁹⁰⁰⁾.

See also properties of table:

- CellBorderColor⁽⁸⁵³⁾;
- BorderColor⁽⁸⁵¹⁾;
- CellBorderStyle⁽⁸⁵⁴⁾.

7.1.4.1.1.7 TRVTableCellData.BorderLightColor

"Light" color of 3d border for this cell

property BorderLightColor: TRVColor⁽⁹⁹⁶⁾;

(introduced in version 1.6)

If this value of this property is *rvcNone*⁽¹⁰³⁸⁾, the cell uses table.CellBorderLightColor⁽⁸⁵⁴⁾.

This value defines light color of 3d border.

A direct assignment to this property cannot be undone/redone by user, use `table.SetCellBorderLightColor`⁽⁸⁸⁶⁾.

Default value:

`rvclNone`⁽¹⁰³⁸⁾

See also properties:

- `BorderColor`⁽⁸⁹⁹⁾;
- `Color`⁽⁹⁰⁰⁾.

See also properties of table:

- `CellBorderColor`⁽⁸⁵³⁾;
- `BorderColor`⁽⁸⁵¹⁾;
- `CellBorderStyle`⁽⁸⁵⁴⁾.

7.1.4.1.1.8 TRVTableCellData.Color

Background color of the cell

property `Color: TRVColor`⁽⁹⁹⁶⁾;

Set to `rvclNone`⁽¹⁰³⁸⁾ 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`⁽⁸⁸⁷⁾.

If this property is equal to `clNone`, a color of a row/column⁽⁸⁶⁷⁾ can be used.

Default value:

`rvclNone`⁽¹⁰³⁸⁾

See also properties:

- `BorderColor`⁽⁸⁹⁹⁾;
- `BorderLightColor`⁽⁸⁹⁹⁾.

See also properties of table:

- `Color`⁽⁸⁵⁸⁾;
- `CellOverrideColor`⁽⁸⁵⁶⁾;
- `CellBorderColor`⁽⁸⁵³⁾;
- `CellBorderLightColor`⁽⁸⁵⁴⁾.

7.1.4.1.1.9 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:

Cells[0,0].ColSpan=3		
Cells[1,0].RowSpan=3		

ColSpan Example

This property corresponds to `<td colspan>` in HTML.

See also:

- [RowSpan](#)⁽⁹⁰⁵⁾.

7.1.4.1.1.10 TRVTableCellData.Hint

Contains the text string that can appear when the user moves the mouse pointer over the cell.

property Hint: TRVUnicodeString⁽¹⁰³²⁾;

(introduced in version 10)

Default value:

" (empty string)

See also:

- [TRVExtraltemStrProperty](#)⁽¹⁰⁰⁵⁾ (*rvespHint* defines items hints, including table hints);
- [TCustomRichView.OnItemHint](#)⁽³⁸¹⁾ (unlike item hints, cell hints cannot be altered in this event).

7.1.4.1.1.11 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*⁽⁸⁹⁸⁾ > 0, it defines the cell height
- if *BestHeight*⁽⁸⁹⁸⁾ = 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 *rvtolgnoreContentHeight* is not included in *table.Options*⁽⁸⁶¹⁾. Otherwise, the table works as if all its cells have **IgnoreContentHeight** = *True*.

Default value:

False

See also:

- [Table Resizing](#)⁽²⁰⁵⁾.

7.1.4.1.1.12 TRVTableCellData.Opacity

Opacity of cell background

property Opacity: TRVOpacity¹⁰¹⁵;

(introduced in version 16)

This value is used if the cell color is *c/None*. 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`⁸⁸⁷.

The cell color is specified in `Color`⁹⁰⁰ property. If `Color`⁹⁰⁰ = *c/None*, the default row/column color⁸⁶⁷ is used.

Default value:

100000 (i.e. 100%)

See also properties of table:

- Opacity⁸⁶⁰

7.1.4.1.1.13 TRVTableCellData.Options

Options for table cell

type

```
TRVTableCellOption = (rvtcoReverseLineOrder, rvtcoUseVerticalFonts);
TRVTableCellOptions = set of TRVTableCellOption;
```

property Options: TRVTableCellOptions;

(introduced in version 16)

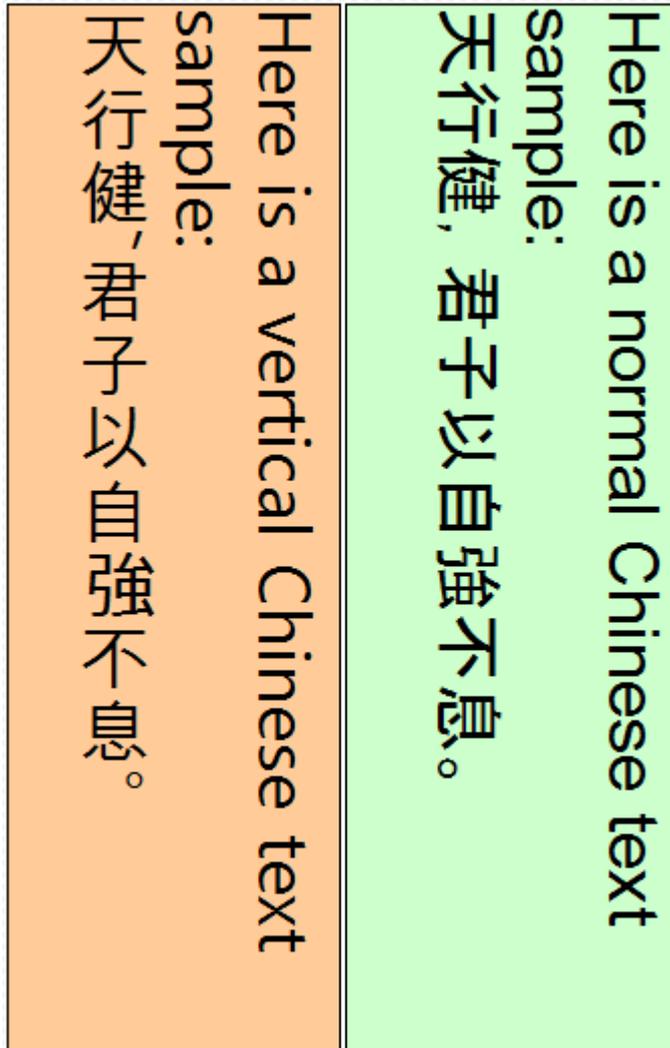
Option	Meaning
<i>rvtcoReverseLineOrder</i>	If set, lines in this cell are have reverse order. For example: <ul style="list-style-type: none"> • if Rotation⁹⁰⁴ = <i>rvrotNone</i>, lines are ordered from bottom to top; • if Rotation⁹⁰⁴ = <i>rvrot90</i>, lines are ordered from left to right
<i>rvtcoUseVerticalFonts</i>	If set, the component adds '@' to the beginning of font names to when applying fonts to canvas, if Rotation ⁹⁰⁴ is <i>rvrot90</i> or <i>rvrot270</i> .

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 Rotation⁹⁰⁴ = *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 Rotation⁹⁰⁴=*rvrot90*, VAlign⁹⁰⁶=*rvcBottom*, Options=[*rvtcoReverseLineOrder*]. In this script, columns are ordered from left to right.

A rotation by 90° can be used to implement vertical text, including East Asian scripts. Use it in conjunction with Options ⁽⁹⁰²⁾.

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 ⁽¹⁶⁸⁾ 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 ⁽¹²⁴⁾;
- 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 ⁽⁹⁰⁶⁾ property;
- Table.CellPadding ⁽⁸⁵⁶⁾

Default value:

rvrotNone

7.1.4.1.1.15 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:

Cells[0,0].ColSpan=3		
Cells[1,0]. RowSpan=3		

RowSpan Example

This property corresponds to <td rowspan> in HTML.

See also:

- ColSpan ⁽⁹⁰⁰⁾.

7.1.4.1.1.16 TRVTableCellData.Tag

Contains additional string value associated with this table cell.

property Tag: TRVTag⁽¹⁰²⁹⁾;

(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⁽⁹¹⁾

7.1.4.1.1.17 TRVTableCellData.VAlign

Defines vertical alignment of the cell content.

property VAlign: TRVCellVAlign⁽⁹⁹⁶⁾;

(introduced in version 1.6)

If this property is set to *rvcVDefault*, the cell uses vertical alignment of its row, see TRVTableRow.VAlign⁽⁹¹⁰⁾.

A direct assignment to this property cannot be undone/redone by user, use `table.SetCellVAlign`⁽⁸⁸⁸⁾.

If the cell is rotated⁽⁹⁰⁴⁾, 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

7.1.4.1.1.18 TRVTableCellData.VisibleBorders

Allows to hide some sides of cell border.

property VisibleBorders: TRVBooleanRect⁽⁹⁶¹⁾;

A direct assignment to this property cannot be undone/redone by user, use `table.SetCellVisibleBorders`⁽⁸⁸⁹⁾.

Default value:

True, True, True, True (all sides are visible)

See also properties of table:

- VisibleBorders⁽⁸⁶⁵⁾;
- CellBorderWidth⁽⁸⁵⁵⁾;
- CellBorderColor⁽⁸⁵³⁾;
- CellBorderLightColor⁽⁸⁵⁴⁾.

7.1.4.1.2 Methods

Inherited from TRVItemFormattedData ⁽⁹⁵⁷⁾

Edit ⁽⁹⁵⁹⁾
 GetOriginEx ⁽⁹⁶⁰⁾
 GetSourceRVData ⁽⁹⁶⁰⁾
 Also read here ⁽⁹⁰⁸⁾

In TRVTableCellData

GetRVData ⁽⁹⁰⁷⁾

7.1.4.1.2.1 TRVTableCellData.GetRVData

Returns object containing cell data (document items)

```
function GetRVData: TCustomRVData (957);
```

The cell is "an object containing data" itself. But when cell is in editing state ⁽¹⁹⁷⁾, 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 ⁽²⁴⁷⁾ property of inplace editor, otherwise.

So when working with content of cell directly ⁽⁹⁰⁸⁾, 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 (957);  
begin  
  Result := '';  
  for r := 0 to table.RowCount (864) - 1 do  
  begin  
    for c := 0 to table.ColCount (858) - 1 do  
    if table.Cells (857)[r,c] <> nil then  
    begin  
      RVData := table.Cells[r,c].GetRVData;  
      for i := 0 to RVData.ItemCount (237) - 1 do  
      begin  
        if (i > 0) and RVData.IsFromNewLine (317)(i) then  
          Result := Result + #13#10;  
        if RVData.GetItemStyle (302)(i) = rvsTab then  
          Result := Result + #9  
        else if RVData.GetItemStyle(i) >= 0 then  
          Result := Result + RVData.GetItemText (303)(i);  
        end;  
      Result := Result + table.TextColSeparator (865);  
    end;  
  Result := Result + table.TextRowSeparator (865);
```

```
end ;
end ;
```

See also methods:

- [GetSourceRVData](#) ⁽⁹⁶⁰⁾.

7.1.4.1.3 Additional properties of TRVTableCellData

[TRVTableCellData](#) ⁽⁸⁹⁵⁾ has many properties and methods, similar to properties and methods of [TRichView](#) ⁽²¹⁰⁾.

Some of them are listed below.

But when the cell is in editing state, [RichViewEdit](#) ⁽⁴⁶¹⁾ 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](#) ⁽⁹⁰⁷⁾ 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 ⁽¹⁹¹⁾.

See also: [TRVTableItemInfo.Changed](#) ⁽⁸⁷¹⁾

Cell methods for appending items

(links are to the corresponding methods of [TCustomRichView](#))

- [AddBreak](#) ⁽²⁶²⁾;
- [AddBullet](#) ⁽²⁶³⁾;
- [AddControl](#) ⁽²⁶⁵⁾;
- [AddHotspot](#) ⁽²⁶⁸⁾;
- [AddFmt](#) ⁽²⁶⁶⁾, [AddItem](#) ⁽²⁶⁹⁾;
- [AddNL](#) ⁽²⁷⁰⁾;
- [AddPicture](#) ⁽²⁷²⁾;

(note: it's not recommended to use the methods of the group above for cells of table inserted in [TRichViewEdit](#) ⁽⁴⁶¹⁾, if editor's [UndoLimit](#) ⁽⁴⁸¹⁾ <>0)

Obtaining information from cell

- [GetItemStyle](#) ⁽³⁰²⁾;
- [GetBreakInfo](#) ⁽²⁸⁹⁾, [GetBulletInfo](#) ⁽²⁸⁹⁾, [GetControlInfo](#) ⁽²⁹³⁾, [GetHotspotInfo](#) ⁽²⁹⁵⁾, [GetPictureInfo](#) ⁽³¹⁰⁾, [GetTextInfo](#) ⁽³¹³⁾;
- [GetItemTag](#) ⁽³⁰²⁾, [IsParaStart](#) ⁽³¹⁸⁾, [GetItemPara](#) ⁽³⁰¹⁾, [IsFromNewLine](#) ⁽³¹⁷⁾, [GetOffsBeforeItem](#) ⁽³⁰⁹⁾, [GetOffsAfterItem](#) ⁽³⁰⁸⁾.

Modifying items

- [SetBreakInfo](#) ⁽³⁵¹⁾, [SetBulletInfo](#) ⁽³⁵²⁾, [SetControlInfo](#) ⁽³⁵⁴⁾, [SetHotspotInfo](#) ⁽³⁵⁷⁾, [SetPictureInfo](#) ⁽³⁶³⁾, [SetItemTag](#) ⁽³⁶⁰⁾.

(note: it's not recommended to use the methods of the group above for cells of table inserted in [TRichViewEdit](#) ⁽⁴⁶¹⁾, if editor's [UndoLimit](#) ⁽⁴⁸¹⁾ <>0)

7.1.4.2 TRVTableRow

This class represents one row in table (TRVTableItemInfo)⁸⁴⁶

Unit RVTable;

Syntax

```
TRVTableRow = class(TRVDataList)
```

Hierarchy

TObject

TList

TRVList

TRVDataList

Using

Table has Rows⁸⁶⁴ property of class TRVTableRows⁹¹¹. 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 TRVTableItemInfo⁸⁴⁶ instead.

Properties

The key properties of this class are Items⁹¹⁰ and Count. Count is the number of columns in the table (all rows in the table have the same number of cells).

VAlign⁹¹⁰ defines the default vertical alignment for the table cells in this row.

PageBreakBefore⁹¹⁰ allows adding a page break before this row.

KeepWithNext⁹¹⁰ allows keeping the content of this row on a single page, if possible.

See Also

Table Cells Overview¹⁹²

7.1.4.2.1 Properties

In TRVTableRow

- Items⁹¹⁰
- KeepTogether⁹¹⁰
- PageBreakBefore⁹¹⁰
- VAlign⁹¹⁰

Derived from TList

Count

7.1.4.2.1.1 TRVTableRow.Items

Provides indexed access to the cells of table

property Items[Index: Integer]:TRVTableCellData⁸⁹⁵; **default**;

Each item represents one cell of the table.

Usually cells are accessed as Cells⁸⁵⁷ 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.PrintOptions⁸⁶³.

The effect is similar to including *rvpaoKeepWithNext* in Options⁷²³ of all paragraphs of all cells of this row.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowKeepTogether⁸⁸⁹.

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.PrintOptions⁸⁶³ is ignored.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowPageBreakBefore⁸⁸⁹.

Default value:

False

7.1.4.2.1.4 TRVTableRow.VAlign

Defines default vertical alignment for contents of cells in the row

property VAlign:TRVCellVAlign⁹⁹⁶;

(introduced in version 1.6)

rvcVDefault works like *rvcMiddle*.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowVAlign⁸⁸⁹.

Default value:

rvcTop

See also properties of cell:

- VAlign⁽⁹⁰⁶⁾

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⁽⁸⁴⁶⁾)

Unit RVTable.

Syntax

```
TRVTableRows = class(TRVList)
```

Hierarchy

TObject

TList

TRVList

Using

This is the type of Rows⁽⁸⁶⁴⁾ properties of table. TRVTableRows is a list of TRVTableRow⁽⁹⁰⁹⁾s which are in their order are lists of TRVTableCellData⁽⁸⁹⁵⁾ (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⁽⁸⁴⁶⁾ instead.

The key properties of this class are Items⁽⁹¹²⁾ and Count. Count is the number of rows in the table.

This class has method for finding main cell for given merged cell: GetMainCell⁽⁹¹²⁾.

See Also

Table Cells Overview⁽¹⁹²⁾

7.1.4.3.1 Properties

In TRVTableRows

- Items⁽⁹¹²⁾

Derived from TList

Count

7.1.4.3.1.1 TRVTableRows.Items

Provides indexed access to the rows of table

```
property Items[Index: Integer]: TRVTableRow909; default;
```

Each item represents one row of the table.

7.1.4.3.2 Methods

In TRVTableRows

GetMainCell⁹¹²

(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): TRVTableCellData
```

Some cells⁸⁵⁷ in table⁸⁴⁶ 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.Cells⁸⁵⁷[**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.Cells⁸⁵⁷[**MRow**, **MCol**]

7.2 Label - TRVLabelItemInfo

TRVLabelItemInfo is a class representing "label"¹⁷⁶ 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 AddItem²⁶⁹) or inserted into TRichViewEdit (see InsertItem⁵²²). Style³⁰² of this item type: *rvsLabel* (-200)

Unit RVLabelItem.

Syntax

```
TRVLabelItemInfo = class(TRVRectItemInfo)
```

(introduced in version 10)

Hierarchy

TObject

TPersistent

*TCustomRVItemInfo*⁸⁴⁴

TRVNonTextItemInfo

TRVRectItemInfo

Properties

This item displays Text⁹¹⁵ using font defined in TextStyleNo⁹¹⁵. ProtectTextStyleNo⁹¹⁴ can be used to protect TextStyleNo⁹¹⁵ from changing.

You can set minimal possible width for this item (MinWidth⁹¹⁴) and horizontal text Alignment⁹¹⁴.

You can define Cursor⁹¹⁴ for this item.

Use SetCurrentItemExtraIntPropertyEx⁵⁵⁶ and SetCurrentItemExtraStrPropertyEx⁵⁵⁷ to change properties of label items as editing operations.

Related Properties of TRVStyle⁶³⁰

- FieldHighlightColor⁶³⁷ – color for highlighting label items (overrides background color⁶⁹⁸ specified in the text style)
- FieldHighlightType⁶³⁸ specifies when to highlight label items.

Inherited Classes

The following item types are inherited from this class:

- TRVSeqItemInfo⁹²¹ (numbered sequences);
- TRVEndNoteItemInfo⁹²⁸ (endnote);
- TRVFootnoteItemInfo⁹³⁰ (footnote);
- TRVNoteReferenceItemInfo⁹⁴⁹ (reference to the parent footnote or endnote).

See Also...

Demos:

- Demos*\Assorted\Fields\MailMerge-LabelItems\

7.2.1 Properties

In TRVLabelItemInfo

Alignment⁹¹⁴

Cursor⁹¹⁴

MinWidth⁹¹⁴

ProtectTextStyleNo⁹¹⁴

RemoveInternalLeading⁹¹⁵

RemoveSpaceBelow⁹¹⁵

Text⁹¹⁵

TextStyleNo⁹¹⁵

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`⁽⁹¹⁵⁾ width is less than `MinWidth`⁽⁹¹⁴⁾.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (`rveipcAlignment`⁽¹⁰⁵¹⁾).

Default value:

taLeftJustify

7.2.1.2 TRVLabelItemInfo.Cursor

Cursor for the item.

```
Cursor: TCursor;
```

Value of this property is not stored in RVF⁽¹²⁴⁾.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (`rveipcCursor`⁽¹⁰⁵¹⁾).

Default value:

crDefault (using default cursor)

7.2.1.3 TRVLabelItemInfo.MinWidth

Defines a minimal width of this item.

```
property MinWidth: TRVStyleLength(1027);
```

This value is measured in Units⁽⁶⁵⁵⁾ of the linked `TRVStyle`⁽⁶³⁰⁾ component.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (`rveipcMinWidth`⁽¹⁰⁵¹⁾).

Default value:

0

See also properties:

- `Alignment`⁽⁹¹⁴⁾;
- `Text`⁽⁹¹⁵⁾.

7.2.1.4 TRVLabelItemInfo.ProtectTextStyleNo

If set to *True*, `TCustomRichViewEdit.ApplyTextStyle`⁽⁴⁹⁴⁾ and `ApplyStyleConversion`⁽⁴⁹²⁾ will not be able to change `TextStyleNo`⁽⁹¹⁵⁾ of this item.

```
ProtectTextStyleNo: Boolean;
```

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (`rveipcProtectTextStyleNo`⁽¹⁰⁵¹⁾).

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`⁹¹⁵ 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 alignment¹⁰³³ = *rvvaLeft*).

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁵⁵⁶ to change values of these properties as an editing operation (*rveipcRemoveInternalLeading* and *rveipcRemoveSpaceBelow*¹⁰⁵¹).

Default value:

- **RemoveInternalLeading**: *False*
- **RemoveSpaceBelow**: *False*
- **UseTypoMetric**: *True*

7.2.1.6 TRVLabelItemInfo.Text

Visible text

```
Text: TRVUnicodeString1032;
```

(changed in version 18)

Unlike text items¹⁶¹, this is not the text returned by `TCustomRichView.GetItemText`³⁰³.

Use `TRichViewEdit.SetCurrentItemExtraStrPropertyEx`⁵⁵⁷ to change value of this property as an editing operation (*RVLabelItem.rvespcText*¹⁰⁵⁶).

See also properties:

- **Alignment**⁹¹⁴;
- **TextStyleNo**⁹¹⁵.

7.2.1.7 TRVLabelItemInfo.TextStyleNo

Defines text attributes.

```
property TextStyleNo: Integer;
```

This is the index in the `TextStyles`⁶⁵⁴ collection of the `TRVStyle`⁶³⁰ component linked to the `TCustomRichView`²¹⁰ component containing this item.

If this item is inserted in editor⁴⁶¹, value of this property may be changed when `ApplyTextStyle`⁴⁹⁴ or `ApplyStyleConversion`⁴⁹² method is called. You can prevent this by assigning *True* to `ProtectTextStyleNo`⁹¹⁴ property.

If style templates²⁵⁶ are used, the component may automatically change value of this property to provide a consistency of text and paragraph style templates (even if `ProtectTextStyleNo`⁹¹⁴=*True*).

See also properties:

- Text⁹¹⁵.

7.2.2 Methods

In TRVLabelItemInfo

Create⁹¹⁶

CreateEx⁹¹⁶

7.2.2.1 TRVLabelItemInfo.Create

A constructor, creates a new TRVLabelItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹, where you wish to insert this item.

This constructor creates a new label with empty Text⁹¹⁵ and TextStyleNo⁹¹⁵=0.

There is more convenient constructor, CreateEx⁹¹⁶.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

7.2.2.2 TRVLabelItemInfo.CreateEx

A constructor, creates a new TRVLabelItemInfo object with the specified Text⁹¹⁵ and TextStyleNo⁹¹⁵.

```
constructor CreateEx(RVData: TPersistent;  
  TextStyleNo: Integer; const Text: TRVUnicodeString1032);
```

(changed in version 18)

Parameters:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹ where you wish to insert this item.

TextStyleNo – value for TextStyleNo⁹¹⁵ property.

Text – value for Text⁹¹⁵ property.

See also methods:

- Create⁹¹⁶.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

7.3 Math expression - TRVMathItemInfo

TRVMathItemInfo is a class representing mathematical expressions⁽¹⁸⁷⁾ (equations, formulas) in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ 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⁽⁸⁴⁴⁾

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⁽⁹¹⁹⁾ property.

By default, the item uses font name, size and color defined in TRVMathDocObject⁽⁹⁷⁴⁾ item of DocObjects⁽²²⁹⁾ collection (or "Cambria Math, 10pt, c/WindowText, if this object does not exist). You can override them in FontName⁽⁹¹⁸⁾, FontSizeDouble⁽⁹¹⁹⁾, TextColor⁽⁹²⁰⁾ properties.

Use `SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ to change properties of mathematical items as editing operations.

7.3.1 Properties

In TRVMathItemInfo

`DisplayInline`⁽⁹¹⁸⁾
`FontName`⁽⁹¹⁸⁾
`FontSizeDouble`⁽⁹¹⁹⁾
`Text`⁽⁹¹⁹⁾
`TextColor`⁽⁹²⁰⁾

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`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (*rveipcDisplayInline*⁽¹⁰⁵¹⁾).

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`⁽⁹⁷⁴⁾ item of `DocObjects`⁽²²⁹⁾ 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`⁽⁵⁵⁷⁾ to change value of this property as an editing operation (*rvespcFontName*⁽¹⁰⁵⁶⁾).

See also:

- `FontSizeDouble`⁽⁹¹⁹⁾

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`⁽⁹⁷⁴⁾ item of `DocObjects`⁽²²⁹⁾ collection.

This value is measured in half-points, i.e. `FontSizeDouble = 20` means 10pt, `FontSizeDouble=21` means 10.5pt.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation (*rveipcFontSizeDouble*⁽¹⁰⁵¹⁾).

See also:

- `FontName`⁽⁹¹⁸⁾

7.3.1.4 TRVMathItemInfo.Text

Mathematical expression in LaTeX-like format.

```
property Text: TRVUnicodeString(1032);
```

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⁽¹⁶¹⁾, this is not the text returned by `TCustomRichView.GetItemText`⁽³⁰³⁾.

Use `TRichViewEdit.SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ to change value of this property as an editing operation (*RVMathItem.rvespcText*⁽¹⁰⁵⁶⁾).

Examples of values and results:

```

\forall x \in X, \quad \exists y \leq \epsilon

```

$$\forall x \in X, \exists y \leq \epsilon$$

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

$$x = a_0 + \frac{1}{\frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}}$$

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

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 *c/None*. This means the item uses the default text color defined in TRVMathDocObject⁽⁸⁰³⁾ item of DocObjects⁽²²⁹⁾ collection.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾ to change value of this property as an editing operation (*rveipcTextColor*⁽¹⁰⁵¹⁾).

Default value:

c/None

7.3.2 Methods

In TRVMathItemInfo

Create⁽⁹²⁰⁾

7.3.2.1 TRVMathItemInfo.Create

A constructor, creates a new TRVMathItemInfo object

constructor Create(RVData: TPersistent); **override**;

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new item with empty Text⁽⁹¹⁹⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.4 Numbered sequence - TRVSeqItemInfo

TRVSeqItemInfo is a class representing numbered sequence⁽¹⁷⁷⁾ in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ of this item type: *rvsSequence* (-202)

Unit RVSeqItem.

Syntax

```
TRVSeqItemInfo = class(TRVLabelItemInfo(912))
```

(introduced in version 10)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo⁽⁹¹²⁾

Properties

The main property of this class is SeqName⁽⁹²⁴⁾. All items having the same value of this property are numbered continuously. You can restart numbering from the specified value, see StartFrom⁽⁹²⁴⁾ and Reset⁽⁹²³⁾ properties.

Numbering type is defined in NumberType⁽⁹²³⁾ property.

Font for this item is defined in TextStyleNo⁽⁹¹⁵⁾. ProtectTextStyleNo⁽⁹¹⁴⁾ can be used to protect TextStyleNo⁽⁹¹⁵⁾ from changing.

Text⁽⁹¹⁵⁾ for this item is calculated automatically.

Use SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾ and SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾ to change properties of label items as editing operations.

Related Properties of TRVStyle⁽⁶³⁰⁾

- FieldHighlightColor⁽⁶³⁷⁾ – color for highlighting numbering sequences (overrides background color⁽⁶⁹⁸⁾ specified in the text style)
- FieldHighlightType⁽⁶³⁸⁾ specifies when to highlight numbering sequences.

Numbering Sequences and List Markers

Numbered sequences are similar to list markers⁽¹⁷⁴⁾. The table below shows differences between them.

Numbered sequences	List markers
Items can be inserted in any place of document.	List markers start paragraphs.

Numbered sequences	List markers
Counter for the item is calculated using SeqName ⁽⁹²⁴⁾ 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 ⁽⁶⁴⁶⁾ .
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 ⁽⁴⁹⁴⁾ and ApplyStyleConversion ⁽⁴⁹²⁾ can change item font.	TCustomRichViewEdit.ApplyTextStyle ⁽⁴⁹⁴⁾ and ApplyStyleConversion ⁽⁴⁹²⁾ cannot change item font.

Inherited Classes

The following item types are inherited from this class:

- TRVEndNoteItemInfo⁽⁹²⁸⁾ (endnote);
- TRVFootnoteItemInfo⁽⁹³⁰⁾ (footnote);

See Also...

Demos:

- Demos*\Assorted\Fields\MailMerge-LabelItems\

7.4.1 Properties

In TRVSeqItemInfo

FormatString⁽⁹²³⁾
 NumberType⁽⁹²³⁾
 Reset⁽⁹²³⁾
 SeqName⁽⁹²⁴⁾
 StartFrom⁽⁹²⁴⁾

Inherited from TRVLabelItemInfo⁽⁹¹²⁾

Alignment⁽⁹¹⁴⁾
 Cursor⁽⁹¹⁴⁾
 MinWidth⁽⁹¹⁴⁾
 ProtectTextStyleNo⁽⁹¹⁴⁾
 Text⁽⁹¹⁵⁾
 TextStyleNo⁽⁹¹⁵⁾

7.4.1.1 TRVSeqItemInfo.FormatString

Text format.

```
FormatString: TRVUnicodeString(1032);
```

(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⁽⁹²³⁾ = *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⁽³³²⁾.

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾ 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(1026);
```

See TRVSeqType⁽¹⁰²⁶⁾ for possible values.

This property is initialized in the constructor CreateEx⁽⁹²⁵⁾.

This property is ignored in inherited classes: TRVFootnoteItemInfo⁽⁹³⁰⁾, TRVEndnoteItemInfo⁽⁹²⁸⁾, TRVSidenoteItemInfo⁽⁹³³⁾, TRVTextBoxItemInfo⁽⁹⁴⁷⁾.

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences⁽³³²⁾.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾ to change value of this property as an editing operation.

See also properties:

- FormatString⁽⁹²³⁾.

7.4.1.3 TRVSeqItemInfo.Reset

Resets counter to StartFrom⁽⁹²⁴⁾.

```
Reset: Boolean;
```

This property is initialized in constructor CreateEx⁽⁹²⁵⁾.

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences⁽³³²⁾.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ 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(1032);
```

(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`⁽⁹³⁰⁾);
- '@endnote@' (used in `TRVEndnoteItemInfo`⁽⁹²⁸⁾).
- '@sidenote@' (used in `TRVSidenoteItemInfo`⁽⁹³³⁾).
- '@textbox@' (used in `TRVTextBoxItemInfo`⁽⁹⁴⁷⁾).

This property is initialized in constructor `CreateEx`⁽⁹²⁵⁾.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation.

7.4.1.5 TRVSeqItemInfo.StartFrom

Value of counter for this item, if `Reset`⁽⁹²³⁾ = `True`.

```
StartFrom: Integer;
```

This property is initialized in constructor `CreateEx`⁽⁹²⁵⁾.

If you change this property for an item that was already inserted in a document, call `RichView.RefreshSequences`⁽³³²⁾.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ to change value of this property as an editing operation.

7.4.2 Methods

In TRVSeqItemInfo

`Create`⁽⁹²⁴⁾

`CreateEx`⁽⁹²⁵⁾

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`⁽⁹²⁵⁾ instead.

7.4.2.2 TRVSeqItemInfo.CreateEx

A constructor, creates a new TRVSeqItemInfo object.

```
constructor CreateEx(RVData: TPersistent;
  const ASeqName: TRVUnicodeString1032;
  ANumberType: TRVSeqType; ATextStyleNo, AStartFrom: Integer;
  AReset: Boolean);
```

(changed in version 18)

Parameters:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹ where you wish to insert this item.

ASeqName – value for SeqName⁹²⁴ property.

ANumberType – value for NumberType⁹²³ property.

ATextStyleNo – value for TextStyleNo⁹¹⁵ property.

AStartFrom – value for StartFrom⁹²⁴ property.

AReset – value for Reset⁹²³ property.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

7.5 Ancestor for notes - TCustomRVNoteItemInfo

TCustomRVNoteItemInfo is an ancestor class for footnote¹⁸⁰, endnote¹⁷⁸, sidenote¹⁸¹ and text box¹⁸³ items.

Objects of this class must not be created directly, create objects of the following inherited classes:

- TRVFootnoteItemInfo⁹³⁰;
- TRVEndnoteItemInfo⁹²⁸;
- TRVSidenoteItemInfo⁹³³;
- TRVTextBoxLayoutItemInfo⁹⁴⁷.

Unit RVNote.

Syntax

```
TCustomRVNoteItemInfo = class(TRVSeqItemInfo921)
TRVFootOrEndnoteItemInfo = class(TCustomRVNoteItemInfo)
```

(introduced in version 10 and 15)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁸⁴⁴

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo ⁹¹²

TRVSeqItemInfo ⁹²¹

Properties

This class introduces Document ⁹²⁶ property – footnote/endnote/sidenote/textbox text. This text is not displayed in TCustomRichView, but it is displayed on printing and in ScaleRichView.

Document ⁹²⁶ uses the same TRVStyle component as the main document (where a note is inserted).

You can use ReplaceDocumentEd ⁹²⁷ method to change Document ⁹²⁵ as an editing operation.

See Also...

Demos:

- Demos*\Editors\Notes

7.5.1 Properties

In TCustomRVNoteItemInfo

- ▶ Document ⁹²⁶
- ▶ NoteText ⁹²⁷

Inherited from TRVSeqItemInfo ⁹²¹

FormatString ⁹²³
 NumberType ⁹²³
 Reset ⁹²³
 SeqName ⁹²⁴
 StartFrom ⁹²⁴

Inherited from TRVLabelItemInfo ⁹¹²

Alignment ⁹¹⁴
 Cursor ⁹¹⁴
 MinWidth ⁹¹⁴
 ProtectTextStyleNo ⁹¹⁴
 Text ⁹¹⁵
 TextStyleNo ⁹¹⁵

7.5.1.1 TCustomRVNoteItemInfo.Document

Footnote/endnote text (displayed on printing)

property Document: TRVNoteData ⁹⁵⁷ ;

You can use ReplaceDocumentEd ⁹²⁷ method to change Document as an editing operation.

7.5.1.2 TCustomRVNoteItemInfo.NoteText

Text for assigning to NoteText⁽²⁴⁰⁾ property of TRichViewEdit⁽⁴⁶¹⁾.

```
property NoteText: TRVUnicodeString(1032);
```

(introduced in version 15; changed in version 18)

This property returns Text⁽⁹¹⁵⁾.

If TRichViewEdit is used as an editor for Document⁽⁹²⁶⁾, you should assign this property to its NoteText⁽²⁴⁰⁾ property, to allow a proper displaying of note references⁽¹⁸⁴⁾.

7.5.2 Methods

In TCustomRVNoteItemInfo

CreateEx⁽⁹²⁷⁾

ReplaceDocumentEd⁽⁹²⁷⁾

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⁽⁹³⁰⁾;
- TRVEndnoteItemInfo⁽⁹²⁸⁾;
- TRVSidenoteItemInfo⁽⁹³³⁾;
- TRVTextBoxLayoutItemInfo⁽⁹⁴⁷⁾.

Parameters

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾ where you wish to insert this item.

ATextStyleNo – value for TextStyleNo⁽⁹¹⁵⁾ property. It's recommended to use RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ function for assigning this parameter.

AStartFrom – value for StartFrom⁽⁹²⁴⁾ property.

AReset – value for Reset⁽⁹²³⁾ property.

7.5.2.2 TCustomRVNoteItemInfo.ReplaceDocumentEd

Replaces content of Document⁽⁹²⁶⁾ as an editing operation.

```
procedure ReplaceDocumentEd(Stream: TStream);
```

Stream must contain new document in RVF format⁽¹²⁴⁾.

7.6 Endnote - TRVEndnoteItemInfo

TRVEndnoteItemInfo is a class representing an endnote⁽¹⁷⁸⁾ in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ of this item type: *rvsEndnote* (-204)

Unit RVNote.

Syntax

```
TRVEndnoteItemInfo = class(TRVFootOrEndnoteItemInfo(925))
```

(introduced in version 10)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo⁽⁹¹²⁾

TRVSeqItemInfo⁽⁹²¹⁾

TCustomRVNoteItemInfo⁽⁹²⁵⁾

TRVFootOrEndnoteItemInfo⁽⁹²⁵⁾

Using

Numbering type for endnotes is defined in EndnoteNumbering⁽⁶³⁷⁾ property of TRVStyle component (NumberType⁽⁹²³⁾ property, inherited from TRVSeqItemInfo⁽⁹²¹⁾, is ignored).

You can enumerate all endnotes in documents using RVGetFirstEndnote and RVGetNextEndnote⁽⁹⁸⁹⁾ functions.

Use RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ to assign TextStyleNo⁽⁹¹⁵⁾ property (in the constructor).

Limitation: in TRVReportHelper⁽⁸⁰⁴⁾, all pages consisting only of endnotes' texts have the same height as the last document's page.

Use SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾ and SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾ to change properties of label items as editing operations.

Example

This example inserts a new endnote in rveMain (TCustomRichViewEdit). TextStyleNo⁽⁹¹⁵⁾ for this endnote is returned by RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ function basing on the rveMain.CurTextStyleNo⁽⁴⁷⁴⁾-th style).

Document⁽⁹²⁶⁾ for this endnote contains one reference⁽¹⁸⁴⁾ (TextStyleNo⁽⁹¹⁵⁾ for this reference is returned by RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ function basing on the 0th style) and one space character of the 0th style.

```
var EndNote: TRVEndnoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo(949);
begin
```

```

EndNote := TRVEndnoteItemInfo.CreateEx930 (rveMain.RVData247 ,
    RVGetNoteTextStyleNo990 (rvs, rveMain.CurTextStyleNo474), 1, False);
NoteRef := TRVNoteReferenceItemInfo949.CreateEx951 (EndNote.Document926 ,
    RVGetNoteTextStyleNo990 (rveMain.Style253, 0));
EndNote.Document926.AddItem269 ('', NoteRef);
EndNote.Document926.AddNL270 ('', 0, -1);
if rveMain.InsertItem522 ('', EndNote) then
    ...
end;

```

The topic about TRVNoteReferenceItemInfo⁹⁴⁹ contains example showing how to edit Document⁹²⁶.

See Also...

Demos:

- Demos*\Editors\Notes

7.6.1 Properties

Inherited from TCustomRVNoteItemInfo⁹²⁵

- ▶ Document⁹²⁶
- ▶ NoteText⁹²⁷

Inherited from TRVSeqItemInfo⁹²¹

FormatString⁹²³
 NumberType⁹²³
 Reset⁹²³
 SeqName⁹²⁴
 StartFrom⁹²⁴

Inherited from TRVLabelItemInfo⁹¹²

Alignment⁹¹⁴
 Cursor⁹¹⁴
 MinWidth⁹¹⁴
 ProtectTextStyleNo⁹¹⁴
 Text⁹¹⁵
 TextStyleNo⁹¹⁵

7.6.2 Methods

In TRVEndnoteItemInfo

Create⁹³⁰
 CreateEx⁹³⁰

Inherited from TCustomRVNoteItemInfo⁹²⁵

ReplaceDocumentEd⁹²⁷

7.6.2.1 TRVEndnoteItemInfo.Create

A constructor, creates a new TRVEndnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new endnote with TextStyleNo⁽⁹¹⁵⁾=0.

There is more convenient constructor, CreateEx⁽⁹³⁰⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

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⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾ where you wish to insert this item.

ATextStyleNo – value for TextStyleNo⁽⁹¹⁵⁾ property. It's recommended to use RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ function for assigning this parameter.

AStartFrom – value for StartFrom⁽⁹²⁴⁾ property.

AReset – value for Reset⁽⁹²³⁾ property.

See also methods:

- Create⁽⁹³⁰⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.7 Footnote - TRVFootnoteItemInfo

TRVFootnoteItemInfo is a class representing a footnote⁽¹⁸⁰⁾ in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ of this item type: *rvsFootnote* (-203)

Unit RVNote.

Syntax

```
TRVFootnoteItemInfo = class(TRVFootOrEndnoteItemInfo925)
```

(introduced in version 10)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁸⁴⁴

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo⁹¹²

TRVSeqItemInfo⁹²¹

TCustomRVNoteItemInfo⁹²⁵

TRVFootOrEndnoteItemInfo⁹²⁵

Using

Numbering type for footnotes is defined in FootnoteNumbering⁶³⁷ property of TRVStyle⁶³⁰ component (NumberType⁹²³ property, inherited from TRVSeqItemInfo⁹²¹, is ignored).

You can enumerate all footnotes in documents using RVGetFirstFootnote and RVGetNextFootnote⁹⁸⁹ functions.

Use RVGetNoteTextStyleNo⁹⁹⁰ to assign TextStyleNo⁹¹⁵ property (in the constructor).

Footnotes are numbered continuously in TCustomRichView. But when printing, footnote numbering can be restarted on each page, see FootnotePageReset⁶⁴⁰ property of TRVStyle⁶³⁰ component.

Limitation: footnote's Document⁹²⁶ must not be higher than the page height. Unlike endnotes¹⁷⁸, footnote is printed on a single page.

Use SetCurrentItemExtraIntPropertyEx⁵⁵⁶ and SetCurrentItemExtraStrPropertyEx⁵⁵⁷ to change properties of label items as editing operations.

Example

This example inserts a new footnote in rveMain (TCustomRichViewEdit). TextStyleNo⁹¹⁵ for this footnote is returned by RVGetNoteTextStyleNo⁹⁹⁰ function basing on the rveMain.CurTextStyleNo⁴⁷⁴ -th style).

Document⁹²⁶ for this footnote contains one reference¹⁸⁴ (TextStyleNo⁹¹⁵ for this reference is returned by RVGetNoteTextStyleNo⁹⁹⁰ function basing on the 0th style) and one space character of the 0th style.

```
var FootNote: TRVFootnoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo949;
begin
    FootNote := TRVFootnoteItemInfo.CreateEx933(rveMain.RVData247,
        RVGetNoteTextStyleNo990(rvs, rveMain.CurTextStyleNo474), 1, False);
    NoteRef := TRVNoteReferenceItemInfo949.CreateEx951(FootNote.Document926,
        RVGetNoteTextStyleNo990(rveMain.Style253, 0));
    FootNote.Document926.AddItem269(' ', NoteRef);
    FootNote.Document926.AddNL270(' ', 0, -1);
```

```

if rveMain.InsertItem522 ('', FootNote) then
    ...
end;

```

The topic about TRVNoteReferenceltemInfo⁹⁴⁹ contains example showing how to edit Document⁹²⁶.

See Also...

Demos:

- Demos*\Editors\Notes

7.7.1 Properties

Inherited from TCustomRVNoteltemInfo⁹²⁵

- ▶ Document⁹²⁶
- ▶ NoteText⁹²⁷

Inherited from TRVSeqItemInfo⁹²¹

FormatString⁹²³
 NumberType⁹²³
 Reset⁹²³
 SeqName⁹²⁴
 StartFrom⁹²⁴

Inherited from TRVLabelItemInfo⁹¹²

Alignment⁹¹⁴
 Cursor⁹¹⁴
 MinWidth⁹¹⁴
 ProtectTextStyleNo⁹¹⁴
 Text⁹¹⁵
 TextStyleNo⁹¹⁵

7.7.2 Methods

In TRVFootnoteltemInfo

Create⁹³²
 CreateEx⁹³³

Inherited from TCustomRVNoteltemInfo⁹²⁵

ReplaceDocumentEd⁹²⁷

7.7.2.1 TRVFootnoteltemInfo.Create

A constructor, creates a new TRVFootnoteltemInfo object

```

constructor Create(RVData: TPersistent); override;

```

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new footnote with TextStyleNo⁽⁹¹⁵⁾=0.

There is more convenient constructor, CreateEx⁽⁹³³⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

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⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾ where you wish to insert this item.

ATextStyleNo – value for TextStyleNo⁽⁹¹⁵⁾ property. It's recommended to use RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ function for assigning this parameter.

AStartFrom – value for StartFrom⁽⁹²⁴⁾ property.

AReset – value for Reset⁽⁹²³⁾ property.

See also methods:

- Create⁽⁹³²⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.8 Sidenote - TRVSidenoteItemInfo

TRVSidenoteItemInfo is a class representing a sidenote⁽¹⁸¹⁾ 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⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ of this item type: *rvsSidenote* (-206)

Unit RVSidenote.

Syntax

```
TRVSidenoteItemInfo = class(TCustomRVNoteItemInfo(925))
```

(introduced in version 15)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo ⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo ⁽⁹¹²⁾

TRVSeqItemInfo ⁽⁹²¹⁾

TCustomRVNoteItemInfo ⁽⁹²⁵⁾

Using

Numbering type for sidenotes is defined in *SidenoteNumbering* ⁽⁶³⁷⁾ property of *TRVStyle* ⁽⁶³⁰⁾ component (*NumberType* ⁽⁹²³⁾ property, inherited from *TRVSeqItemInfo* ⁽⁹²¹⁾, is ignored).

You can enumerate all sidenotes in documents using *RVGetFirstSidenote* and *RVGetNextSidenote* ⁽⁹⁸⁹⁾ functions (these functions return text box items ⁽¹⁸³⁾ as well)

Use *RVGetNoteTextStyleNo* ⁽⁹⁹⁰⁾ to assign *TextStyleNo* ⁽⁹¹⁵⁾ property (in the constructor).

This class introduces the following properties:

- *BoxPosition* ⁽⁹³⁵⁾ – contains properties defining the position of the floating box
- *BoxProperties* ⁽⁹³⁵⁾ – contains properties describing size, background and border of the floating box.

Use *SetCurrentItemExtraIntPropertyEx* ⁽⁵⁵⁶⁾ and *SetCurrentItemExtraStrPropertyEx* ⁽⁵⁵⁷⁾ to change properties of sidenotes as editing operations.

Example

This example inserts a new sidenote in *rveMain* (*TCustomRichViewEdit*). *TextStyleNo* ⁽⁹¹⁵⁾ for this sidenote is returned by *RVGetNoteTextStyleNo* ⁽⁹⁹⁰⁾ function basing on the *rveMain.CurTextStyleNo* ⁽⁴⁷⁴⁾-th style).

Document ⁽⁹²⁶⁾ for this sidenote contains one reference ⁽¹⁸⁴⁾ (*TextStyleNo* ⁽⁹¹⁵⁾ for this reference is returned by *RVGetNoteTextStyleNo* ⁽⁹⁹⁰⁾ 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 (949);
begin
    SideNote := TRVSidenoteItemInfo.CreateEx (933) (rveMain.RVData (247),
        RVGetNoteTextStyleNo (990) (rvs, rveMain.CurTextStyleNo (474)), 1, False);
    NoteRef := TRVNoteReferenceItemInfo (949).CreateEx (951) (SideNote.Document (926),
        RVGetNoteTextStyleNo (990) (rveMain.Style (253), 0));
    SideNote.Document (926).AddItem (269) (' ', NoteRef);
    SideNote.Document (926).AddNL (270) (' ', 0, -1);
    SideNote.BoxProperties (935).Background (938).Color (742) := clYellow;
    SideNote.BoxProperties (935).Border (938).Style (744) := rvbDouble;
```

```

SideNote.BoxPosition935.VerticalOffset946 :=
    rveMain.Style253.PixelsToUnits659(50);
if rveMain.InsertItem522(' ', SideNote) then
    ...
end;

```

The topic about TRVNoteReferenceItemInfo⁹⁴⁹ contains example showing how to edit Document⁹²⁶.

See also

- TRVTextBoxItemInfo⁹⁴⁷

7.8.1 Properties

In TRVSidenoteItemInfo

BoxPosition⁹³⁵
BoxProperties⁹³⁵

Inherited from TCustomRVNoteItemInfo⁹²⁵

- ▶ Document⁹²⁶
- ▶ NoteText⁹²⁷

Inherited from TRVSeqItemInfo⁹²¹

FormatString⁹²³
NumberType⁹²³
Reset⁹²³
SeqName⁹²⁴
StartFrom⁹²⁴

Inherited from TRVLabelItemInfo⁹¹²

Alignment⁹¹⁴
Cursor⁹¹⁴
MinWidth⁹¹⁴
ProtectTextStyleNo⁹¹⁴
Text⁹¹⁵
TextStyleNo⁹¹⁵

7.8.1.1 TRVSidenoteItemInfo.BoxProperties

Properties of a floating box: size, border, background, vertical alignment of content.

```
property BoxProperties: TRVBoxProperties937;
```

7.8.1.2 TRVSidenoteItemInfo.BoxPosition

Properties of a floating box: horizontal and vertical positions, z-order.

```
property BoxPosition: TRVBoxPosition940;
```

7.8.2 Methods

In TRVSidenoteItemInfo

Create⁹³⁶

CreateEx⁹³⁶

Inherited from TCustomRVNoteItemInfo⁹²⁵

ReplaceDocumentEd⁹²⁷

7.8.2.1 TRVSidenoteItemInfo.Create

A constructor, creates a new TRVSidenoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹, where you wish to insert this item.

This constructor creates a new sidenote with TextStyleNo⁹¹⁵=0.

There is more convenient constructor, CreateEx⁹³⁶.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

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 – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹ where you wish to insert this item.

ATextStyleNo – value for TextStyleNo⁹¹⁵ property. It's recommended to use RVGetNoteTextStyleNo⁹⁹⁰ function for assigning this parameter.

AStartFrom – value for StartFrom⁹²⁴ property.

AReset – value for Reset⁹²³ property.

See also methods:

- Create⁹³⁶.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- `InsertItem`⁽⁵²²⁾.

7.8.3 Classes of properties

Classes of `TRVSidenoteItemInfo`⁽⁹³³⁾ Properties

- `TRVBoxProperties`⁽⁹³⁷⁾ defines the text box size and appearance. This is a type of `BoxProperties`⁽⁹³⁵⁾ property.
- `TRVBoxPosition`⁽⁹⁴⁰⁾ defines the text box position. This is a type of `BoxPosition`⁽⁹³⁵⁾ 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`⁽⁹³⁵⁾ property of sidenotes⁽¹⁸¹⁾ and text boxes⁽¹⁸³⁾.

Properties:

- `Background`⁽⁹³⁸⁾;
- `Border`⁽⁹³⁸⁾;
- `Height`, `HeightType`⁽⁹³⁸⁾;
- `Width`, `WidthType`⁽⁹³⁹⁾;
- `VAlign`⁽⁹³⁹⁾.

7.8.3.1.1 Properties

In `TRVBoxProperties`

`Background`⁽⁹³⁸⁾
`Border`⁽⁹³⁸⁾
`Height`⁽⁹³⁸⁾
`HeightType`⁽⁹³⁸⁾
`Width`⁽⁹³⁹⁾
`WidthType`⁽⁹³⁹⁾
`VAlign`⁽⁹³⁹⁾

7.8.3.1.1.1 TRVBoxProperties.Background

Defines background properties.

type

```
TRVBoxBackgroundRect = class (TRVBackgroundRect741)
```

```
property Background: TRVBoxBackgroundRect;
```

TRVBoxBackgroundRect changes the initial values of properties introduced in TRVBackgroundRect⁷⁴¹:

- Color⁷⁴²=*c/Window*
- BorderOffsets⁷⁴²: Left=Right=10, Top=Bottom=5.

Unlike in TRVBackgroundRect, in TRVBoxBackgroundRect negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx⁵⁵⁶ and SetCurrentItemExtraStrPropertyEx⁵⁵⁷ 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 (TRVBorder742)
```

```
property Border: TRVBoxBorder;
```

TRVBoxBorder changes the initial values of properties introduced in TRVBorder⁷⁴²:

- Style⁷⁴⁴=*rvbSingle*
- BorderOffsets⁷⁴³: Left=Right=10, Top=Bottom=5.

Unlike in TRVBorder, in TRVBoxBorder negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx⁵⁵⁶ and SetCurrentItemExtraStrPropertyEx⁵⁵⁷ 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: TRVFloatSize1007;
```

Value of HeightType	Meaning
<i>rvbhtAbsolute</i>	Height specifies the box height in Units ⁶⁵⁵ of TRVStyle
<i>rvbhtRelPage</i>	Height specifies the box height in (1/1000)% of the page height
<i>rvbhtRelMainTextArea</i>	Height specifies the box height in (1/1000)% of the main text area height

Value of HeightType	Meaning
<i>rvbhtRelTopMargin</i>	Height specifies the box height in (1/1000)% of the top margin height
<i>rvbhtRelBottomMargin</i>	Height specifies the box height in (1/1000)% of the bottom margin height
<i>rvbhtAuto</i>	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`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ 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⁽⁹³⁸⁾ `<>rvbhtAuto`.

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ to change value of this property as an editing operation.

Default value:

t/Top

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: TRVFloatSize1007;
```

Value of WidthType	Meaning
<i>rvbwtAbsolute</i>	Width specifies the box width in Units ⁶⁵⁵ of TRVStyle
<i>rvbwtRelPage</i>	Width specifies the box width in (1/1000)% of the page width
<i>rvbwtRelMainTextArea</i>	Width specifies the box width in (1/1000)% of the main text area width
<i>rvbwtRelLeftMargin</i>	Width specifies the box width in (1/1000)% of the left margin width
<i>rvbwtRelRightMargin</i>	Width specifies the box width in (1/1000)% of the right margin width
<i>rvbwtRelInnerMargin</i>	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)
<i>rvbwtRelOuterMargin</i>	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.SetCurrentItemExtraIntPropertyEx556` and `SetCurrentItemExtraStrPropertyEx557` 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`⁹³⁵ property of `sidenotes`¹⁸¹ and text boxes¹⁸³.

Properties:

- `HorizontalAnchor`⁹⁴¹ and `HorizontalPositionKind`, `HorizontalOffset`, `HorizontalAlignment`⁹⁴²;
- `VerticalAnchor`⁹⁴⁵ and `VerticalPositionKind`, `VerticalOffset`, `VerticalAlignment`⁹⁴⁶;
- `RelativeToCell`⁹⁴⁵;
- `PositionInText`⁹⁴⁴.

7.8.3.2.1 Properties

In `TRVBoxPosition`

`HorizontalAlignment`⁹⁴²
`HorizontalAnchor`⁹⁴¹
`HorizontalOffset`⁹⁴²
`HorizontalPositionKind`⁹⁴²
`PositionInText`⁹⁴⁴
`RelativeToCell`⁹⁴⁵
`VerticalAlignment`⁹⁴⁶
`VerticalAnchor`⁹⁴⁵
`VerticalOffset`⁹⁴⁶
`VerticalPositionKind`⁹⁴⁶

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`⁷⁵⁹, a left margin starts from the page's left edge and ends at `Margins`⁷⁶⁸.Left.

If `RelativeToCell`⁹⁴⁵ = `True`, and a `sidenote`¹⁸¹ (or a text box item¹⁸³) is inserted in a table cell¹⁹², then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
<code>rvhanCharacter</code>	A <code>sidenote</code> ¹⁸¹ character, or a place of insertion of a text box item ¹⁸³ .	

Value	Anchor Area	Scheme
<i>rvhanPage</i>	A whole page (or a whole table cell)	
<i>rvhanMainTextArea</i>	A main text area, i.e. an area between margins (or a table cell area minus cellpadding)	
<i>rvhanLeftMargin</i>	A left page margin (or a left cell padding area)	
<i>rvhanRightMargin</i>	A right page margin (or a right cell padding area)	
<i>rvhanInnerMargin</i>	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)	
<i>rvhanOuterMargin</i>	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`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ 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(1006);
property HorizontalOffset: TRVFloatPosition(1006);
property HorizontalAlignment: TRVFloatHorizontalAlignment;
```

A position is calculated relative to the area specified in `HorizontalAnchor`⁽⁹⁴¹⁾.

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`⁽⁹⁴¹⁾.

Note 2: printers cannot print at areas close to paper edges; if 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
<i>rvfphalLeft</i>	The left side of the box is at the left side of the anchor area.	
<i>rvfphalCenter</i>	The center of the box is at center of the anchor area.	
<i>rvfphalRight</i>	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⁽⁶⁵⁵⁾ of the linked TRVStyle⁽⁶³⁰⁾.

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`⁵⁵⁶ and `SetCurrentItemExtraStrPropertyEx`⁵⁵⁷ to change values of these properties as an editing operation.

Default values:

- `HorizontalPositionKind`: *rvfpkAbsPosition*
- `HorizontalOffset`: 0
- `HorizontalAlignment`: *rvfpalLeft*

7.8.3.2.1.3 TRVBoxPosition.PositionInText

Specifies how the text box is positioned relative to a text where a sidenote¹⁸¹ (or a text box item¹⁸³) is inserted.

type

```
TRVFloatPositionInText = (rvpitAboveText, rvpitBelowText);
```

property `PositionInText`: `TRVFloatPositionInText`;

Value	Meaning
<i>rvpitAboveText</i>	The box is above a text.
<i>rvpitBelowText</i>	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`⁵⁵⁶ and `SetCurrentItemExtraStrPropertyEx`⁵⁵⁷ 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⁽¹⁹²⁾.

property `RelativeToCell`: Boolean;

If *True*, and a sidenote⁽¹⁸¹⁾ (or a text box item⁽¹⁸³⁾) is placed in a table cell, then horizontal⁽⁹⁴¹⁾ and vertical⁽⁹⁴⁵⁾ positions are calculated relative to this cell (instead of a page).

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ 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`⁽⁷⁵⁹⁾, a top margin starts from the page's top edge and ends at `Margins`⁽⁷⁶⁸⁾.`Top`.

If `RelativeToCell`⁽⁹⁴⁵⁾ = *True*, and a sidenote⁽¹⁸¹⁾ (or a text box item⁽¹⁸³⁾) is inserted in a table cell⁽¹⁹²⁾, then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
<i>rvvanLine</i>	A line where a sidenote ⁽¹⁸¹⁾ character (or a text box item ⁽¹⁸³⁾) is inserted.	
<i>rvvanParagraph</i>	A paragraph where a sidenote ⁽¹⁸¹⁾ character (or a text box item ⁽¹⁸³⁾) is inserted.	
<i>rvvanPage</i>	A whole page (or a whole table cell)	
<i>rvvanMainTextArea</i>	A main text area, i.e. an area between margins (or a table cell area minus cellpadding)	
<i>rvvanTopMargin</i>	A top page margin (or a top cell padding area)	
<i>rvvanBottomMargin</i>	A bottom page margin (or a bottom cell padding area)	

Use `TRichViewEdit.SetCurrentItemExtraIntPropertyEx`⁽⁵⁵⁶⁾ and `SetCurrentItemExtraStrPropertyEx`⁽⁵⁵⁷⁾ 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:

rvanLine

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: TRVFloatPositionKind1006;
property VerticalOffset: TRVFloatPosition1006;
property VerticalAlignment: TRVFloatVerticalAlignment;
```

A position is calculated relative to the area specified in VerticalAnchor⁹⁴⁵.

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 VerticalAnchor⁹⁴⁵.

Note 2: printers cannot print at areas close to paper edges; if 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
<i>rvfpvalTop</i>	The top side of the box is at the top side of the anchor area.
<i>rvfpvalCenter</i>	The center of the box is at center of the anchor area.
<i>rvfpvalBottom</i>	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 ⁽⁶⁵⁵⁾ of the linked TRVStyle ⁽⁶³⁰⁾.

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 ⁽⁵⁵⁶⁾ and SetCurrentItemExtraStrPropertyEx ⁽⁵⁵⁷⁾ 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 ⁽¹⁸³⁾ 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 ⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem ⁽⁵²²⁾). Style ⁽³⁰²⁾ of this item type: *rvtTextBox* (-207)

Unit RVSideNote.

Syntax

```
TRVTextBoxItemInfo = class(TRVSideNoteItemInfo (933))
```

(introduced in version 15)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo ⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo ⁽⁹¹²⁾

TRVSeqItemInfo ⁽⁹²¹⁾

TCustomRVNoteItemInfo ⁽⁹²⁵⁾

TRVSideNoteItemInfo ⁽⁹³³⁾

Using

This is a simplified version of a sidenote⁽¹⁸¹⁾. It has all properties of a sidenote, but the most of them are ignored, except for `BoxPosition`⁽⁹³⁵⁾ and `BoxProperties`⁽⁹³⁵⁾.

A special mark can be drawn in places of insertion of text box items, see `TRVStyle.FloatingLineColor`⁽⁶³⁸⁾.

7.9.1 Methods

In TRVSidenoteItemInfo

`Create`⁽⁹⁴⁸⁾
`CreateEx`⁽⁹⁴⁸⁾

Inherited from TCustomRVNoteItemInfo⁽⁹²⁵⁾

`ReplaceDocumentEd`⁽⁹²⁷⁾

7.9.1.1 TRVTextBoxItemInfo.Create

A constructor, creates a new `TRVFootnoteItemInfo` object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – `RVData`⁽²⁴⁷⁾ property of `TRichView`⁽²¹⁰⁾ or `TRichViewEdit`⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new text box item with `TextStyleNo`⁽⁹¹⁵⁾=0.

See also `CreateEx`⁽⁹⁴⁸⁾.

See also methods of TCustomRichView:

- `AddItem`⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- `InsertItem`⁽⁵²²⁾.

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`⁽⁹⁴⁸⁾.

Parameters:

RVData – `RVData`⁽²⁴⁷⁾ property of `TRichView`⁽²¹⁰⁾ or `TRichViewEdit`⁽⁴⁶¹⁾ where you wish to insert this item.

ATextStyleNo – ignored.

AStartFrom – value for `StartFrom`⁽⁹²⁴⁾ property. Assigned, but not used in this type of item.

AReset – value for Reset⁽⁹²³⁾ property. Assigned, but not used in this type of item.

See also methods:

- Create⁽⁹⁴⁸⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.10 Reference to the parent note - TRVNoteReferenceItemInfo

TRVNoteReferenceItemInfo is an item type referring to⁽¹⁸⁴⁾ the parent footnote⁽¹⁸⁰⁾, endnote⁽¹⁷⁸⁾, or sidenote⁽¹⁸¹⁾ item. Objects of this class are usually appended to the note's Document⁽⁹²⁶⁾ (see AddItem⁽²⁶⁹⁾). Besides, they can be appended to TRichView (see AddItem⁽²⁶⁹⁾) or inserted into TRichViewEdit (see InsertItem⁽⁵²²⁾). Style⁽³⁰²⁾ of this item type: *rvsNoteReference* (-205).

Unit RVNote.

Syntax

```
TRVNoteReferenceItemInfo = class(TRVLabelItemInfo(912))
```

(introduced in version 10)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁽⁸⁴⁴⁾

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo⁽⁹¹²⁾

Using

When inserted in a note's Document⁽⁹²⁶⁾, the note reference displays the same text as the parent note.

When inserted in TCustomRichView, it displays NoteText⁽²⁴⁰⁾.

Text⁽⁹¹⁵⁾ for this item is calculated automatically.

Use RVGetNoteTextStyleNo⁽⁹⁹⁰⁾ to assign TextStyleNo⁽⁹¹⁵⁾ property (in the constructor).

Use SetCurrentItemExtraIntPropertyEx⁽⁵⁵⁶⁾ and SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾ to change properties of label items as editing operations.

Example

See examples in TRVFootnoteItemInfo⁽⁹³⁰⁾ and TRVEndNoteItemInfo⁽⁹²⁸⁾ showing how to insert footnote/endnote with document containing note reference.

This example shows how to edit note document in rveNote (TCustomRichViewEdit⁴⁶¹). The footnote/endnote is NoteItem.

This code copies document from footnote/endnote to the editor:

```
var Stream: TMemoryStream;
begin
    rveNote.NoteText240 := NoteItem.Text915;
    rveNote.Clear279;
    Stream := TMemoryStream.Create;
    try
        NoteItem.Document926.SaveRVFToStream344(Stream);
        Stream.Position := 0;
        rveNote.LoadRVFFromStream321(Stream);
    finally
        Stream.Free;
    end;
    with rveNote do begin
        Format288;
        SetSelectionBounds365(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 NoteText²⁴⁰. 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.Modified479 then
    exit;
Stream := TMemoryStream.Create;
try
    rveNote.SaveRVFToStream344(Stream, False);
    NoteItem.ReplaceDocumentEd927(Stream);
finally
    Stream.Free;
end;
rveNote.Modified479 := False;
```

See Also...

Demos:

- Demos*\Editors\Notes

7.10.1 Properties

Inherited from TRVLabelItemInfo⁹¹²

- Alignment⁹¹⁴
- Cursor⁹¹⁴
- MinWidth⁹¹⁴
- ProtectTextStyleNo⁹¹⁴
- Text⁹¹⁵

TextStyleNo⁹¹⁵

7.10.2 Methods

In TRVNoteReferenceltemInfo

Create⁹⁵¹

CreateEx⁹⁵¹

7.10.2.1 TRVNoteReferenceltemInfo.Create

A constructor creating a new TRVNoteReferenceltemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹, where you wish to insert this item.

This constructor creates a new note reference with TextStyleNo⁹¹⁵=0.

There is more convenient constructor, CreateEx⁹⁵¹.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

7.10.2.2 TRVNoteReferenceltemInfo.CreateEx

A constructor creating a new TRVNoteReferenceltemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; TextStyleNo: Integer);
```

Parameters:

RVData – RVData²⁴⁷ property of TRichView²¹⁰ or TRichViewEdit⁴⁶¹ where you wish to insert this item.

TextStyleNo – value for TextStyleNo⁹¹⁵ property. It's recommended to use RVGetNoteTextStyleNo⁹⁹⁰ function for assigning this parameter.

See also methods:

- Create⁹⁵¹.

See also methods of TCustomRichView:

- AddItem²⁶⁹.

See also methods of TCustomRichViewEdit:

- InsertItem⁵²².

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:

- TRVPageNumberItemInfo⁹⁵³;
- TRVPageCountItemInfo⁹⁵⁴.

Unit RVFieldItems.

Syntax

```
TCustomRVFieldItemInfo = class (TRVLabelItemInfo912);
TCustomRVPageNumberItemInfo = class(TCustomRVFieldItemInfo);
```

(introduced in version 15)

Hierarchy

TObject
 TPersistent
 TCustomRVItemInfo⁸⁴⁴
 TRVNonTextItemInfo
 TRVRectItemInfo
 TRVLabelItemInfo⁹¹²

TCustomRVPageNumberItemInfo introduces a new property: NumberType⁹⁵².

7.11.1 Properties

In TCustomRVPageNumberItemInfo

NumberType⁹⁵²

Inherited from TRVLabelItemInfo⁹¹²

Alignment⁹¹⁴
 Cursor⁹¹⁴
 MinWidth⁹¹⁴
 ProtectTextStyleNo⁹¹⁴
 Text⁹¹⁵
 TextStyleNo⁹¹⁵

7.11.1.1 TCustomRVPageNumberItemInfo.NumberType

Specifies how the page number (or page count) is displayed.

```
NumberType: TRVPageNumberType1016;
```

This property is initialized in the constructor.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx⁵⁵⁶ to change value of this property as an editing operation.

Default value:

rvpntDefault

7.12 Page Number - TRVPageNumberItemInfo

TRVPageNumberItemInfo displays the current page number ¹⁸⁵.

Unit RVFieldItems.

Syntax

```
TRVPageNumberItemInfo = class(TCustomRVPageNumberItemInfo 952)
```

(introduced in version 15)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo ⁸⁴⁴

TRVNonTextItemInfo

TRVRectItemInfo

TRVLabelItemInfo ⁹¹²

Using

When inserted in TRichView ²¹⁰, it is displayed as Text ⁹¹⁵ (by default, as '{p}').

When printed with TRVPrint ⁷⁵⁹, it displays a number of the page where this item is located. Page numbers are counted from TRVPrint.PageNoFromNumber ⁸²⁶.

When drawing with TRVReportHelper ⁸⁰⁴, the default behavior is like in TRVPrint, but you can provide a value for the page number in OnGetPageNumber ⁸¹¹ event.

 **ScaleRichView**: when inserted in TScaleRichViewEdit, it displays a number of the page where this item is located. Page numbers are counted from TScaleRichViewEdit.PageProperty.PageNoFromNumber.

A displayed page number is formatted according to NumberType ⁹⁵² property.

7.12.1 Methods

In TRVPageNumberItemInfo

Create ⁹⁵³

CreateEx ⁹⁵⁴

7.12.1.1 TRVPageNumberItemInfo.Create

A constructor creating a new TRVPageNumberItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new page number field with TextStyleNo⁽⁹¹⁵⁾=0, Text⁽⁹¹⁵⁾='{p}'.

There is more convenient constructor, CreateEx⁽⁹⁵⁴⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.12.1.2 TRVPageNumberItemInfo.CreateEx

A constructor creating a new TRVPageNumberItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType(1016);
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾ where you wish to insert this item.

NumberType – value for NumberType⁽⁹⁵²⁾ property.

TextStyleNo – value for TextStyleNo⁽⁹¹⁵⁾ property.

This constructor creates a new page number field with Text⁽⁹¹⁵⁾='{p}'.

See also methods:

- Create⁽⁹⁵³⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.13 Page Count - TRVPageCountItemInfo

TRVPageCountItemInfo displays the count of pages⁽¹⁸⁶⁾.

Unit RVFieldItems.

Syntax

```
TRVPageCountItemInfo = class(TCustomRVPageNumberItemInfo(952))
```

(introduced in version 15)

Hierarchy

TObject

TPersistent

TCustomRVItemInfo⁽⁸⁴⁴⁾

TRVNonTextItemInfo
 TRVRectItemInfo
 TRVLabelItemInfo⁽⁹¹²⁾

Using

When inserted in TRichView⁽²¹⁰⁾, it is displayed as Text⁽⁹¹⁵⁾ (by default, as '{P}').

When printed with TRVPrint⁽⁷⁵⁹⁾, it displays a count of pages.

 **ScaleRichView**: when inserted in TScaleRichViewEdit, it displays a count of pages.

A displayed page number is formatted according to NumberType⁽⁹⁵²⁾ property.

7.13.1 Methods

In TRVPageCountItemInfo

Create⁽⁹⁵⁵⁾

CreateEx⁽⁹⁵⁵⁾

7.13.1.1 TRVPageCountItemInfo.Create

A constructor creating a new TRVPageCountItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾, where you wish to insert this item.

This constructor creates a new page count field with TextStyleNo⁽⁹¹⁵⁾=0, Text⁽⁹¹⁵⁾='{P}'.

There is more convenient constructor, CreateEx⁽⁹⁵⁵⁾.

See also methods of TCustomRichView:

- AddItem⁽²⁶⁹⁾.

See also methods of TCustomRichViewEdit:

- InsertItem⁽⁵²²⁾.

7.13.1.2 TRVPageCountItemInfo.CreateEx

A constructor creating a new TRVPageCountItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType(1016);
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData⁽²⁴⁷⁾ property of TRichView⁽²¹⁰⁾ or TRichViewEdit⁽⁴⁶¹⁾ where you wish to insert this item.

NumberType – value for NumberType⁽⁹⁵²⁾ property.

TextStyleNo – value for TextStyleNo⁽⁹¹⁵⁾ property.

This constructor creates a new page count field with `Text915='{P}'`.

See also methods:

- `Create955`.

See also methods of TCustomRichView:

- `AddItem269`.

See also methods of TCustomRichViewEdit:

- `InsertItem522`.

8 Other classes

Classes providing uniform VCL and FireMonkey programming interface

- `TRVCanvas961` – TCanvas wrapper.
- `TRVGraphic970` – [VCL/LCL] TGraphic; [FMX] TBitmap wrapper.
- `TRVPicture975` – [VCL/LCL] TPicture; [FMX] TPicture analog.
- `TRVGraphicClass971` – [VCL/LCL] TGraphicClass; [FMX] class of TBitmap.

Other TRichView classes

- `ERichViewError957` – an exception class.
- `TCustomRVData957` – TRichView documents.
- `TPrintableRV961` – a TRichView object used internally by `TRVPrint759`.
- `TRVBooleanRect961` – (Left, Top, Right, Bottom) boolean properties.
- `TRVDeleteUnusedStylesData969` allows deleting unused styles in multiple documents.
- `TRVGraphicHandler972` contains methods for working with graphics.
- `TRVIntegerList974` – a list of integer values.
- `TRVMathDocObject974` contains properties common for all equation objects¹⁸⁷ in a document.
- `TRVRect975` – (Left, Top, Right, Bottom) `TRVStyleLength1027` properties.
- `TRVReportHelperWithHeaderFooters976` contains several `TRVReportHelper804` components representing a main document, headers and footers.
- `TRVStringList978` – a string list that stores Unicode strings
- `TRVThumbnailMaker979` provides high-quality image scaling in TRichView.
- `TRVUnitsRect980` – (Left, Top, Right, Bottom) `TRVLength1015` properties.
- `TRVControlsPainter962` 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 = svcsSimple`.

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)

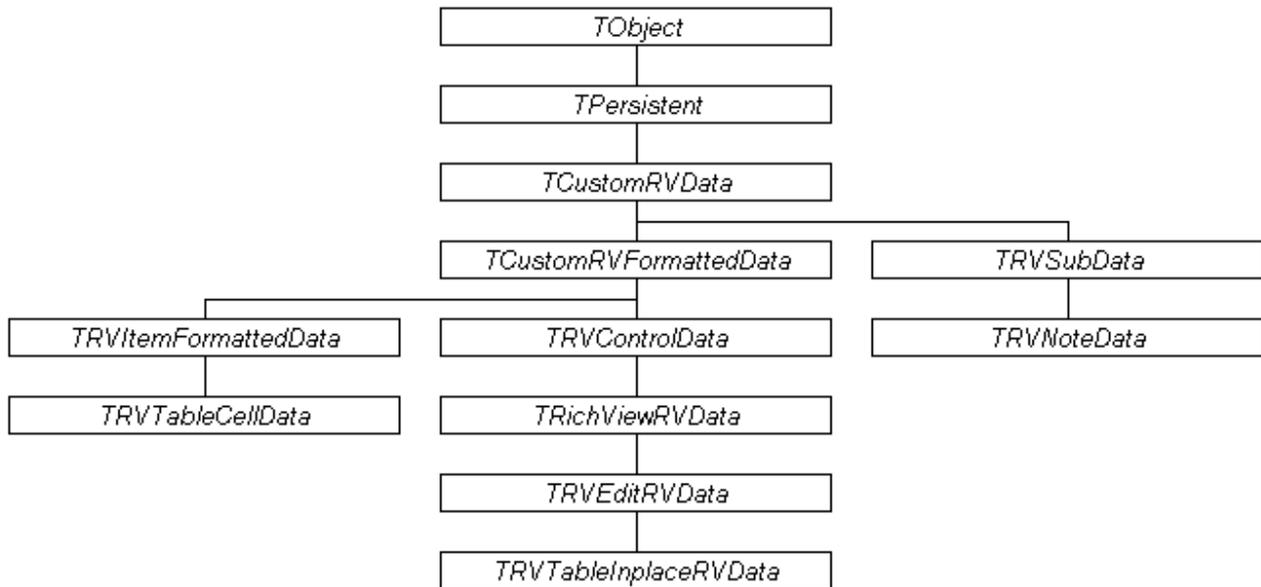
These classes represent RichView documents. Documents contain a list of items. Some items may, in each order, contain subdocuments.

Class	Unit*	Meaning
TCustomRVData	CRVData	Basic class for TRichView documents.
TCustomRVFormattedData	CRVFDData	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 ⁽²¹⁰⁾ and TDBRichView ⁽⁵⁹⁴⁾ , a class of RVData ⁽²⁴⁷⁾ property of these components.
TRVEditRVData	RVERVData	Document in TRichViewEdit ⁽⁴⁶¹⁾ and TDBRichViewEdit ⁽⁶⁰⁶⁾ , a class of RVData ⁽²⁴⁷⁾ property of these components.
TRVTInplaceRVData	RVTInplace	Document in cell inplace editor, a class of of RVData ⁽²⁴⁷⁾ property of cell inplace editors.
TRVItemFormattedData	RVDataList	Basic class for formatted subdocuments owned by items.
TRVTableCellData ⁽⁸⁹⁵⁾	RVTable	Table ⁽¹⁸⁹⁾ 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](#)⁹⁵⁹;
- [GetRVData](#)⁹⁶⁰;
- [GetSourceRVData](#)⁹⁶⁰.

Special methods for **TCustomRVFormattedData**:

- [GetOriginEx](#)⁹⁶⁰.

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](#)²⁶²,
- [AddBullet](#)²⁶³,
- [AddControl](#)²⁶⁵,
- [AddHotspot](#)²⁶⁸,
- [AddFmt](#)²⁶⁶, [AddItem](#)²⁶⁹,
- [AddNL](#)²⁷⁰, [AddTextNL](#)²⁷⁴,
- [AddPicture](#)²⁷².

Obtaining information from cell:

- [GetItemStyle](#)³⁰², [GetItemText](#) -A -W³⁰³,
- [GetBreakInfo](#)²⁸⁹, [GetBulletInfo](#)²⁸⁹, [GetControlInfo](#)²⁹³, [GetHotspotInfo](#)²⁹⁵, [GetPictureInfo](#)³¹⁰
- [GetItemTag](#)³⁰², [GetItemVAlign](#)³⁰³, [IsParaStart](#)³¹⁸, [GetItemPara](#)³⁰¹, [IsFromNewLine](#)³¹⁷

Modifying items:

- `SetBreakInfo`⁽³⁵¹⁾, `SetBulletInfo`⁽³⁵²⁾, `SetControlInfo`⁽³⁵⁴⁾, `SetHotspotInfo`⁽³⁵⁷⁾, `SetPictureInfo`⁽³⁶³⁾, `SetItemTag`⁽³⁶⁰⁾, `SetItemVAlign`⁽³⁶¹⁾, `SetItemText -A -W`⁽³⁶⁰⁾

See also

- `Controls`, `Documents`, `Items`⁽¹³⁶⁾

8.2.1 TCustomRVData.Edit

Initializes editor for this RVData.

function `Edit`: TCustomRVData;

For RVData⁽²⁴⁷⁾ of editor (main editor, or cell inplace editor), this method does nothing.

For table cell⁽⁸⁹⁵⁾, this method creates cell inplace editor and returns its RVData⁽²⁴⁷⁾.

If you want to select part of document in table cell, call `Cell.Edit` (or `table.EditCell`⁽⁸⁷⁶⁾) before calling `cell.SetSelectionBounds`⁽³⁶⁵⁾ or `cell.SelectAll`⁽³⁴⁸⁾.

This method must be called when the document is formatted.

Return value

GetRVData⁽⁹⁶⁰⁾ of the given RVData after initialization of editing.

Example

This example deletes all controls from `rve: TRichViewEdit`⁽⁴⁶¹⁾ 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`⁽²¹⁰⁾. `TCustomRVFormattedData` has all these methods and properties, with the same meaning.

```
procedure DeleteAllControls(RVData: TCustomRVData(957);
  rve: TCustomRichViewEdit(461));
var i,r,c: Integer;
  table: TRVTableItemInfo(846);
begin
  i := 0;
  while i<RVData.ItemCount(237) do begin
    case RVData.GetItemStyle(302)(i) of
      rvsComponent(1059):
        begin
          RVData := RVData.Edit;
          TCustomRVFormattedData(RVData).SetSelectionBounds(365)(i, 0, i, 1);
          rve.DeleteSelection(503);
          i := rve.TopLevelEditor(481).CurItemNo(472)-1;
        end;
      rvsTable(1059):
        begin
          table := TRVTableItemInfo(846)(RVData.GetItem(296)(i));
          for r := 0 to table.RowCount(864)-1 do
```

```

    for c := 0 to table.ColCount858-1 do
        if table.Cells857[r,c]<>nil then
            DeleteAllControls(table.Cells857[r,c].GetRVData907, rve);
        end;
    end; // of case
    inc(i);
end; // of while
end;

Call:
DeleteAllControls(rve.RVData247, 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 (TRVTableCellData⁸⁹⁵). When the cell is not edited, Cell.GetRVData = Cell. But when the cell is edited, this method returns RVData²⁴⁷ of the cell inplace editor.

Using RVData.GetRVData instead of RVData you can always access the document's items.

See also

- GetSourceRVData⁹⁶⁰;
- Edit⁹⁵⁹.

8.2.3 TCustomRVData.GetSourceRVData

This is an inverse method to GetRVData⁹⁶⁰.

```
function GetSourceRVData: TCustomRVData;
```

For the most of document classes, this method returns Self, i.e. RVData.GetSourceRVData = RVData.

The only exception is RVData²⁴⁷ of the cell inplace editor, it returns the source cell (TRVTableCellData⁸⁹⁵) for this editor.

See also

- GetRVData⁹⁶⁰.

8.2.4 TCustomRVFormattedData.GetOriginEx

Returns coordinates of the given subdocument relative to the main document.

```
procedure GetOriginEx(out ALeft, ATop: TRVCoord998);
```

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`⁽²⁹⁸⁾ 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`⁽⁷⁴⁴⁾;
- `TRVTableCellData.VisibleBorders`⁽⁹⁰⁶⁾;
- `TRVTableItemInfo.VisibleBorders`⁽⁸⁶⁵⁾.

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⁽¹⁰⁵¹⁾. 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 color

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 TRVDefaultLoadProperties

This class defines default properties for loading HTML, Markdown, RTF, DocX documents in TRichView⁽²¹⁰⁾.

This is a type of global variable RVDefaultLoadProperties⁽¹⁰⁵¹⁾.

Unit [VCL/FMX]: RVDefReadProps / fmxRVDefReadProps.

Syntax

```
TRVDefaultLoadProperties = class;
```

(introduced in version 20)

Hierarchy

TObject

8.7.1 Properties

In TRVReportHelperWithHeaderFooters

DefaultBulletFontName⁽⁹⁶³⁾
DefaultFontName⁽⁹⁶⁴⁾
DefaultFontSizeDouble⁽⁹⁶⁴⁾
DefaultFontSizesDouble⁽⁹⁶⁴⁾
DefaultMonoSpacedFontName⁽⁹⁶⁵⁾
DefaultProperties⁽⁹⁶⁵⁾
DefaultUnicodeSymbolFontName⁽⁹⁶⁵⁾
GenericFontNames⁽⁹⁶⁵⁾
ListMarkerIndentPix⁽⁹⁶⁸⁾

8.7.1.1 TRVDefaultLoadProperties.DefaultBulletFontName

Default font for bullets (paragraph list markers)

```
property DefaultUnicodeSymbolFontName: TFontName;
```

This font is used by default for paragraph bullets loaded from:

- HTML
- Markdown
- RTF and DocX, if RTFReadProperties⁽²⁴⁶⁾.ConvertSymbolFonts⁽⁴⁵³⁾ = True.

This property provides access to GenericFontNames⁽⁹⁶⁵⁾ [rvgffRVBullets].

8.7.1.2 TRVDefaultLoadProperties.DefaultFontName

Default HTML font.

property DefaultFontName: TFontName;

This property provides access to GenericFontNames⁹⁶⁵ [rvgffRVDefault].

8.7.1.3 TRVDefaultLoadProperties.DefaultFontSizeDouble

Defines fonts size for HTML import, in half-points.

property DefaultFontSizeDouble: Integer;

This property provides access to DefaultFontSizesDouble⁹⁶⁴ [rvfstDefault]

8.7.1.4 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 	CSS font-size	Default Value
<i>rvfstDefault</i>	Used if not defined		24
<i>rvfstXXSmall</i>	1	xx-small	16
<i>rvfstXSmall</i>	–	x-small	18
<i>rvfstSmall</i>	2	small	20
<i>rvfstMedium</i>	3	medium	24
<i>rvfstLarge</i>	4	large	28
<i>rvfstXLarge</i>	5	x-large	36
<i>rvfstXXLarge</i>	6	xx-large	48
<i>rvfstXXXLarge</i>	7	xxx-large	56

It's highly recommend to set `DefaultFontSizeDouble[rvfstDefault]` equal to `DefaultFontSizeDouble[rvfstMedium]`.

8.7.1.5 TRVDefaultLoadProperties.DefaultMonoSpacedFontName

Default monospaced HTML and Markdown font.

property `DefaultMonoSpacedFontName`: `TFontName`;

This property provides access to `GenericFontNames`⁽⁹⁶⁵⁾ [`rvgffUIMonospace`].

8.7.1.6 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(996);
```

```
  Style: TFontStyles;
```

```
end;
```

property `DefaultProperties` [

Index: `TRVFormattingType`]: `TRVFormattingDefaultItem`;

if `StyleTemplates` are not used⁽²⁵⁶⁾, or there are no necessary style templates, this property defines attributes of headings and hyperlinks.

8.7.1.7 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`⁽²⁴⁶⁾.`ConvertSymbolFonts`⁽⁴⁵³⁾ = `True`.

This property provides access to `GenericFontNames`⁽⁹⁶⁵⁾ [`rvgffRVSymbols`].

8.7.1.8 TRVDefaultLoadProperties.GenericFontNames

Default font names.

type

```
TRVGenericFontFamily = (
  rvgffRVDefault, rvgffSerif, rvgffSansSerif,
  rvgffMonospace, rvgffCursive, rvgffFantasy,
  rvgffSystemUI, rvgffUISerif, rvgffUISansSerif,
  rvgffUIMonospace, rvgffUIRounded, rvgffEmoji,
  rvgffMath, rvgffFangsong, rvgffRVSymbols,
```

```
rvgffRVBullets);
```

property GenericFontNames[

Index: TRVGenericFontFamily]: TFontName;

HTML import

The following fonts are used when importing HTML files.

Value	Used
<i>rvgffRVDefault</i>	If font is not defined
<i>rvgffSerif</i>	For CSS "serif" font family
<i>rvgffSansSerif</i>	For CSS "sans-serif" font family
<i>rvgffMonospace</i>	For CSS "monospace" font family (and for tags like <code>)
<i>rvgffCursive</i>	For CSS "cursive" font family
<i>rvgffFantasy</i>	For CSS "fantasy" font family
<i>rvgffSystemUI</i>	For CSS "system-ui" font family
<i>rvgffUISansSerif</i>	For CSS "ui-sans-serif" font family
<i>rvgffUISerif</i>	For CSS "ui-serif" font family
<i>rvgffUIMonospace</i>	For CSS "ui-monospace" font family
<i>rvgffUIRounded</i>	For CSS "ui-rounded" font family
<i>rvgffEmoji</i>	For CSS "emoji" font family
<i>rvgffMath</i>	For CSS "math" font family
<i>rvgffFangsong</i>	For CSS "fangsong" font family
<i>rvgffRVBullets</i>	For bullets (paragraph list markers)

Markdown

The following fonts are used when importing Markdown files.

Value	Used
<i>rvgffMonospace</i>	For code fragments, if StyleTemplates are not used ⁽²⁵⁶⁾ , or there is no 'HTML Code' style template.
<i>rvgffRVBullets</i>	For bullets (paragraph list markers)

RTF and DocX

The following fonts are used when importing RTF and DocX files.

Value	Used
<i>rvgffRVSymbols</i>	Instead of symbol fonts, if <code>ConvertSymbolFonts⁽⁴⁵³⁾ = True</code>
<i>rvgffRVBullets</i>	For bullets (paragraph list markers) instead of symbol fonts, if <code>ConvertSymbolFonts⁽⁴⁵³⁾ = True</code>

Default values

Default values depend on the platform.

Value	Default Value				
	Windows	macOS	iOS	Linux	Android
<i>rvgffRVDefault</i>	'Tahoma'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
<i>rvgffSerif</i>	'Times New Roman'	'Times'	'Times New Roman'	'Liberation Serif'	'serif'
<i>rvgffSansSerif</i>	'Arial'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
<i>rvgffMonospace</i>	'Courier New'	'Courier New'	'Courier New'	'DejaVu Sans Mono'	'monospace'
<i>rvgffCursive</i>	'Comic Sans MS'	'Apple Chancery'	'Snell Roundhand'	'Z003'	'normal'
<i>rvgffFantasy</i>	'Impact'	'Papyrus'	'Papyrus'	'Impact'	'normal'
<i>rvgffSystemUI</i>	'Segoe UI'	'Helvetica'	'Helvetica'	'Liberation Sans'	'normal'
<i>rvgffUISansSerif</i>	Same as for <i>rvgffSansSerif</i>				
<i>rvgffUISerif</i>	Same as for <i>rvgffSerif</i>				
<i>rvgffUIMonospace</i>	Same as for <i>rvgffMonospace</i>				
<i>rvgffUIRounded</i>	Same as for <i>rvgffSansSerif</i>				
<i>rvgffEmoji</i>					
<i>rvgffMath</i>					

<i>rvgffFangsong</i>	'FangSong'	'FangSong'	'FangSong'	'FangSong'	'FangSong'
<i>rvgffRVSymbols</i>	'Segoe UI Symbol'	'Apple Symbols'	'Apple Symbols'	'Liberation Sans'	'normal'
<i>rvgffRVBullets</i>	Same as for <i>rvgffRVDefault</i>				

8.7.1.9 TRVDefaultLoadProperties.ListMarkerIndentPix

A list marker indent for bullets and numbering imported from HTML and Markdown.

property ListMarkerIndentPix: TRVPixel96Length¹⁰¹⁸;

This is the default distance between the left side of a list marker and the left side of its paragraph text.

Default value:

24

8.8 TRVCustomMathEquationManager

This class defines global properties for math expression items¹⁸⁷ in all documents.

Only one object of this class can exist in the application; it is returned by RVMathEquationManager⁹⁹¹ 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:

- SaveMathMLInHTML⁹⁶⁹ enables saving math items as equation objects (MathML format) in HTML files.
- SaveOMMLInDocX⁹⁶⁹ enables saving math items as equation objects (OMML format) in DocX files.

8.8.1 Properties

In TRVReportHelperWithHeaderFooters

SaveMathMLInHTML⁹⁶⁹

SaveOMMLInDocX⁹⁶⁹

8.8.1.1 TRVCustomMathEquationManager.SaveMathMLInHTML

Instructs saving math expressions ⁽¹⁸⁷⁾ 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

8.8.1.2 TRVCustomMathEquationManager.SaveOMMLInDocX

Instructs saving math expressions ⁽¹⁸⁷⁾ 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

8.9 TRVDeleteUnusedStylesData

This class allows to delete unused text ⁽⁶⁵⁴⁾, paragraph ⁽⁶⁴⁸⁾ and list ⁽⁶⁴⁶⁾ 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 ⁽¹²⁴⁾ (but stored in file or in the Registry). In this case, several documents share the same collections of styles. TCustomRichView.DeleteUnusedStyles ⁽²⁸⁶⁾ cannot be used in this case, MarkStylesInUse ⁽³³¹⁾ and DeleteMarkedStyles ⁽²⁸⁴⁾ 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(331)(data);
RichView2.MarkStylesInUse(331)(data);
RichView1.DeleteMarkedStyles(284)(data);
RichView2.DeleteMarkedStyles(284)(data);
data.Free;
```

Example 2

How to remove unused styles in several documents.

```
uses CRVData, RVItem;

data := TRVDeleteUnusedStylesData.Create(True, True, True);
RichView1.LoadRVF(328)('Doc1.rvf');
RichView1.MarkStylesInUse(331)(data);
RichView1.LoadRVF(328)('Doc2.rvf');
RichView1.MarkStylesInUse(331)(data);
...
RichViewDoNotCheckRVFStyleRefs(1045) := True;
RichView1.LoadRVF(328)('Doc1.rvf');
RichView1.DeleteMarkedStyles(284)(data);
RichView1.SaveRVF(344)('Doc1.rvf', False);
RichView1.LoadRVF(328)('Doc2.rvf');
RichView1.DeleteMarkedStyles(284)(data);
RichView1.SaveRVF(344)('Doc2.rvf', False);
...
RichViewDoNotCheckRVFStyleRefs(1045) := False;
data.Free;
```

8.10 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: TRVGraphicType¹⁰¹⁰

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 RVGraphicHandler¹⁰⁵⁷.GetFileDialogFilter⁹⁷² 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.

8.11 TRVGraphicClass

A class type for graphic object.

Unit [VCL/FMX]: RVClasses / fmxRVClasses;

Syntax [VCL]:

```
TRVGraphicClass = TGraphicClass;
```

Syntax [FMX]:

```
TRVGraphicClass = class of TRVGraphicFM970;
```

8.12 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⁽¹⁰⁵⁷⁾. 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(1010);
  GraphicClass: TRVGraphicClass(971));
procedure RegisterPngGraphic(GraphicClass: TRVGraphicClass(971));
procedure RegisterJpegGraphic(GraphicClass: TRVGraphicClass(971));
procedure RegisterHTMLGraphicFormat(GraphicClass: TRVGraphicClass(971));
function CreateGraphicByType(GraphicType: TRVGraphicType(1010)): TRVGraphic(970);
function LoadFromFile(const FileName: TRVUnicodeString(1032)): TRVGraphic(970);
function LoadFromStream(Stream: TStream): TRVGraphic(970);
function GetDataGraphicType(const Buffer; Size: Integer): TRVGraphicType(1010);

// VCL and LCL
function CreateGraphic(GraphicClass: TRVGraphicClass(971)): TRVGraphic(970);
// 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 from TRVGraphicFM⁽⁹⁷⁰⁾ (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⁽²⁴⁵⁾ property (*rvtfPNGInsteadOfBitmap*).

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⁽¹⁵⁴⁾, 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⁽¹⁵⁴⁾;
- jpeg: TJPEGImage;
- png: TPngImage (for Delphi 2009+), TPortableNetworkGraphic (for Lazarus⁽¹⁵⁴⁾), 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⁽¹⁵⁴⁾), 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⁽¹²⁷⁾ 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`⁽⁹⁷⁰⁾ `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`⁽¹⁰⁶⁸⁾

8.13 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

8.14 TRVMathDocObject

This is a class of a document object defining default properties for items-mathematical expressions⁽¹⁸⁷⁾.

Unit `RVMathItem`;

Syntax

```
TRVMathDocObject = class (TRVDocObject(229));
```

(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: `c/WindowText`.

Items may override these properties in their `FontName`⁹¹⁸, `FontSizeDouble`⁹¹⁹, `TextColor`⁹²⁰ properties.

See the explanations about mathematical fonts in the help topics about `TRVMathItemInfo.FontName`⁹¹⁸

How to use

It's not recommended to use this object directly. Use `RVGetMathProperties` and `RVSetMathProperties`⁹⁸⁶ procedures instead.

To save and load these properties in `RVF`¹²⁴, include `rvfoSaveDocObjects` and `rvfoLoadDocObjects` in `RVFOptions`²⁴⁷.

8.15 TRVPicture

A class for objects that contains `TRVGraphic`⁹⁷⁰ 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`⁹⁷⁰ container.

In FireMonkey, `TRVPictureFM` is implemented as an analog of `TPicture`, to allow using the same graphic properties.

See also:

- `TRVStyle`⁶³⁰.`InvalidPicture`⁶⁴³
- `TRVListLevel`⁷⁵⁰.`Picture`⁷⁵⁹

8.16 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⁽¹⁰²⁷⁾ type.

8.17 TRVReportHelperWithHeaderFooters

This class contains TRVReportHelper⁽⁸⁰⁴⁾ 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⁽⁸⁰⁴⁾ component used for the main document (it is accessible as MainDoc⁽⁹⁷⁷⁾)
- TRVReportHelper components used for all six headers and footers (they are accessible as GetHeader⁽⁹⁷⁸⁾ and GetFooter⁽⁹⁷⁸⁾)
- TRVStyle⁽⁶³⁰⁾ components for each TRVReportHelper.

See details in the help topic about the constructor⁽⁹⁷⁷⁾.

Use AssignToRVPrint⁽⁹⁷⁷⁾ to initialize TRVPrint⁽⁷⁵⁹⁾ 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**)

8.17.1 Properties

In TRVReportHelperWithHeaderFooters

- ▶ MainDoc⁽⁹⁷⁷⁾
- UseStyleTemplates⁽⁹⁷⁷⁾

8.17.1.1 TRVReportHelperWithHeaderFooters.MainDoc

Returns TRVReportHelper component used to store the main document.

```
property MainDoc: TRVReportHelper804;
```

See also

- `GetHeader`⁹⁷⁸
- `GetFooter`⁹⁷⁸

8.17.1.2 TRVReportHelperWithHeaderFooters.UseStyleTemplates

Allows or disallows using `MainDoc`⁹⁷⁷.`RichView`⁸⁰⁷.`Style`²⁵³.`StyleTemplates`⁶⁵².

```
property UseStyleTemplates: Boolean;
```

This property provides access to `UseStyleTemplates`²⁵⁶ property of all `RichViews` of all `TRVReportHelpers` stored in the component.

Default value

False

8.17.2 Methods

In TRVReportHelperWithHeaderFooters

```
AssignToRVPrint977  
Create977  
GetFooter978  
GetHeader978  
Reset978
```

8.17.2.1 TRVReportHelperWithHeaderFooters.AssignToRVPrint

Prepares the document stored in this component to printing by **RVPrint**.

```
procedure AssignToRVPrint(RVPrint: TRVPrint759);
```

This procedure prepares **RVPrint** by calling its `AssignSource`⁷⁷¹, `SetHeader`⁷⁷⁹, `SetFooter`⁷⁷⁸, `AssignDocParameters`⁷⁷¹ methods.

This procedure does not format `RVPrint` (it does not call its `FormatPages`⁷⁷⁴).

This procedure does not assign page size (only margins and header&footer positions)

8.17.2.2 TRVReportHelperWithHeaderFooters.Create

Creates an instance of `TRVReportHelperWithHeaderFooters`⁹⁷⁶ component.

```
constructor Create;
```

The constructor:

- creates `TRVReportHelper`⁸⁰⁴ component that will be used for the main document (it is accessible as `MainDoc`⁹⁷⁷)
- creates `TRVReportHelper` components that will be used for all six headers and footers (they are accessible as `GetHeader`⁹⁷⁸ and `GetFooter`⁹⁷⁸), assigns them as headers/footers to the main document;

- creates TRVStyle⁽⁶³⁰⁾ 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⁽⁶⁴⁷⁾ properties to MainDoc⁽⁹⁷⁷⁾.RichView⁽⁸⁰⁷⁾.Style⁽²⁵³⁾ (so, if you decide to use style templates⁽⁹⁷⁷⁾, they all will use MainDoc⁽⁹⁷⁷⁾.RichView⁽⁸⁰⁷⁾.Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾).

Default value

False

8.17.2.3 TRVReportHelperWithHeaderFooters.GetFooter

Returns TRVReportHelper component representing a footer of the specified type.

```
function GetFooter(FooterType: TRVHFTType(1011)): TRVReportHelper(804);
```

See also

- GetHeader⁽⁹⁷⁸⁾
- MainDoc⁽⁹⁷⁷⁾

8.17.2.4 TRVReportHelperWithHeaderFooters.GetHeader

Returns TRVReportHelper component representing a header of the specified type.

```
function GetHeader(FooterType: TRVHFTType(1011)): TRVReportHelper(804);
```

See also

- GetFooter⁽⁹⁷⁸⁾
- MainDoc⁽⁹⁷⁷⁾

8.17.2.5 TRVReportHelperWithHeaderFooters.Reset

Completely clears all documents stored in this component.

```
procedure Reset;
```

This procedure clears⁽²⁷⁹⁾ all documents (the main document, headers, footers), deletes all text⁽⁶⁵⁴⁾, paragraph⁽⁶⁴⁸⁾ and list⁽⁶⁴⁶⁾ styles from all TRVStyles, adds one text and one text and paragraph style having default properties.

If UseStyleTemplates⁽⁹⁷⁷⁾ = True, it also clears StyleTemplates⁽⁶⁵²⁾ for the main document, adds a single normal style template⁽⁶⁸⁹⁾, and assign it to paragraph styles.

Additionally, it assigns 5 pixels to all margins (LeftMargin⁽²³⁸⁾, RightMargin⁽²⁴⁴⁾, TopMargin⁽²⁵⁵⁾, BottomMargin⁽²²⁵⁾), and resets DocParameters⁽²³⁰⁾ (keeping their Units⁽⁴²⁰⁾).

8.18 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 handling duplicates. However, the following methods work and perform correct comparisons: Sort, FindUnicodeString, IndexOfUnicodeString, IndexOfUnicodeName.

Most important methods:

```
function AddUnicodeString(  
    const S: TRVUnicodeString(1032)): Integer;  
function GetUnicodeString(  
    Index: Integer): TRVUnicodeString(1032);  
procedure SetUnicodeString(Index: Integer;  
    const Value: TRVUnicodeString(1032));
```

8.19 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⁽¹⁰⁶¹⁾. 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(970)): Boolean;  
function MakeThumbnail(Gr: TRVGraphic(970); 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 ⁽¹²⁰⁾

8.20 TRVUnitsRect

This class contains 4 properties of TRVLength ⁽¹⁰¹⁵⁾ type: Left, Top, Right, Bottom

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

```
TRVFloatRect = class (TPersistent)
```

(Introduced in version 15)

This class is used for Margins ⁽⁷⁶⁸⁾ property of TRVPrint ⁽⁷⁵⁹⁾.

9 Procedures and functions

TRichView procedures and functions

DrawControl ⁽⁹⁸¹⁾ (and other functions) draws controls on the specified canvas [VCL and LCL]

GetRVESearchOptions ⁽⁹⁸⁶⁾ converts options of a find/search dialog to TRichViewEdit search options [VCL and LCL]

GetRVSearchOptions ⁽⁹⁸⁷⁾ converts options of a find/search dialog to TRichView search options [VCL and LCL]

RV_GetPrinterDC ⁽⁹⁸⁸⁾ creates a DC compatible with the active printer [VCL and LCL]

RVGetFirstEndnote ⁽⁹⁸⁹⁾ (and other functions) enumerates notes in the document

RVGetNoteTextStyleNo ⁽⁹⁹⁰⁾ returns a text file for footnotes or endnotes

RVisURL, RVisEmail ⁽⁹⁹¹⁾ check whether the string contains an URL or e-mail address

`RVOpenURL`⁽⁹⁹¹⁾ opens the specified URL in an external browser

Unicode conversion⁽⁹⁹²⁾ function convert text to and from Unicode

Functions from `RVGetText` unit⁽⁹⁸³⁾ returns text representation of the document or its fragment

Functions from `RVLinear` unit⁽⁹⁸⁴⁾ provide 1:1 conversion between the document and a text string

Functions from `RVMathItem` unit⁽⁹⁸⁶⁾ help to work with common properties of equation objects⁽¹⁸⁷⁾

`RVMathEquationManager`⁽⁹⁹¹⁾ returns the object containing global properties used by equation objects⁽¹⁸⁷⁾

Functions for `TRVUnits` conversion⁽⁹⁸¹⁾

Functions for extraction HTML from RTF⁽⁹⁸⁷⁾

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⁽¹⁶⁸⁾ for default printing of inserted controls⁽¹⁶⁸⁾, see `OnPrintComponent`⁽⁸³¹⁾ event.

9.2 Functions for TRVUnits conversion

unit `[VCL/FMX] RVFuncs / fmxRVFuncs`;

```
function RV_UnitsPerInch(Units: TRVUnits(1032);  
  Control: TControl): TRVLength(1015); overload;
```

```
function RV_UnitsPerInch(Units: TRVUnits(1032);  
  PixelsPerInch: Integer): TRVLength(1015); overload;
```

```
function RV_UnitsToTwips(Value: TRVLength(1015); Units: TRVUnits(1032);  
  Control: TControl): Integer; overload; inline;
```

```
function RV_UnitsToTwips(Value: TRVLength(1015); Units: TRVUnits(1032);  
  PixelsPerInch: Integer): Integer; overload; inline;
```

```

function RV_UnitsToEMU(Value: TRVLength1015; Units: TRVUnits1032;
  Control: TControl): Integer; overload; inline;
function RV_UnitsToEMU(Value: TRVLength1015; Units: TRVUnits1032;
  PixelsPerInch: Integer): Integer; overload; inline;

function RV_UnitsToPixels(Value: TRVLength1015; Units: TRVUnits1032;
  Control: TControl): Integer; overload; inline;
function RV_UnitsToPixels(Value: TRVLength1015; Units: TRVUnits1032;
  PixelsPerInch: Integer): TRVCoord998; overload; inline;

function RV_TwipsToUnits(Value: Integer; Units: TRVUnits1032;
  Control: TControl): TRVLength1015; overload; inline;
function RV_TwipsToUnits(Value: Integer; Units: TRVUnits1032;
  PixelsPerInch: Integer): TRVLength1015; overload; inline;

function RV_EMUToUnits(Value: Integer; Units: TRVUnits1032;
  Control: TControl): TRVLength1015; overload; inline;
function RV_EMUToUnits(Value: Integer; Units: TRVUnits1032;
  PixelsPerInch: Integer): TRVLength1015; overload; inline;

function RV_UnitsToUnits(Value: TRVLength1015;
  OldUnits, NewUnits: TRVUnits1032;
  Control: TControl): TRVLength1015; overload; inline;
function RV_UnitsToUnits(Value: TRVLength1015;
  OldUnits, NewUnits: TRVUnits1032;
  PixelsPerInch: Integer): TRVLength1015; 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 RichViewPixelsPerInch¹⁰⁴⁷ 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⁽¹³⁹⁾

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(461);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetVisibleText(rv: TCustomRichView(210);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetAllText(rv: TCustomRichView(210);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetRVDataText(RVData: TCustomRVData(957);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetCurrentParaSectionText(rve: TCustomRichViewEdit(461);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetCurrentParaText(rve: TCustomRichViewEdit(461);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetCurrentChar(rve: TCustomRichViewEdit(461);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
function GetCurrentWord(rve: TCustomRichViewEdit(461);
    CodePage: TRVCodePage = CP_ACP): TRVAnsiString(993);
```

Unit [VCL/FMX] RVGetTextW / fmxRVGetTextW:

```
function GetCurrentLineText(rve: TCustomRichViewEdit(461);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetVisibleText(rv: TCustomRichView(210);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetAllText(rv: TCustomRichView(210);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetRVDataText(RVData: TCustomRVData(957);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetCurrentParaSectionText(rve: TCustomRichViewEdit(461);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetCurrentParaText(rve: TCustomRichViewEdit(461);
    ForANSI: Boolean = False): TRVUnicodeString(1032);
function GetCurrentChar(rve: TCustomRichViewEdit(461)): TRVUnicodeString(1032);
function GetCurrentWord(rve: TCustomRichViewEdit(461)): TRVUnicodeString(1032);
```

(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`⁹⁸⁴

9.4 Functions from RVLinear Unit

Unit [VCL / FMX] `RVLinear` / `fmRVLinear`.

const

```
RVCharsPerLineBreak: Integer = 2;
RVNonTextCharacter: TRVUnicodeChar = ' ';
```

```
function RVGetLinearCaretPos(rve: TCustomRichViewEdit461;
  CharsPerLineBreak: Integer = -1): Integer;
```

```
procedure RVSetLinearCaretPos(rve: TCustomRichViewEdit461;
  LinearPos: Integer; CharsPerLineBreak: Integer = -1);
```

RVGetLinearCaretPos returns the caret position (as a number of characters before the caret). Non-text objects (except for tables¹⁷³ and tabulators¹⁶²) 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

Value of CharsPerLineBreak parameter	Meaning
1	Use #13 to separate lines
0	Lines are not separated
-1	Use the value of RVCharsPerLineBreak global variable (which, in this order, may be 2, 1, or 0)

```

procedure RVGetSelection(rv: TCustomRichView210;
  out SelStart, SelLength: Integer;
  CharsPerLineBreak: Integer = -1);
procedure RVSetSelection(rv: TCustomRichView210;
  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: TCustomRichView210;
  out Selection: TRVSelection;
  CharsPerLineBreak: Integer = -1);
procedure RVSetSelectionEx(rv: TCustomRichView210;
  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: TCustomRichView210;
  CurRVData, RVData: TCustomRVData957;
  ItemNo, ItemOffs: Integer; out LinearPos: Integer;
  CharsPerLineBreak: Integer = -1): Boolean;
function LinearToRichView(rv: TCustomRichView210;
  CurRVData: TCustomRVData957;
  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 tables¹⁷³ and tabulators¹⁶²) are treated as one character, line breaks are treated as CharsPerLineBreak characters. **LinearToRichView** performs an opposite conversion.

```
function RVGetTextRange(rv: TCustomRichView210;
  RangeStart, RangeLength: Integer;
  CharsPerLineBreak: Integer = -1): TRVUnicodeString1032;
function RVGetTextLength(rv: TCustomRichView210;
  CharsPerLineBreak: Integer = -1): Integer;
```

RVGetTextRange returns text from *rv*, from *RangeStart* to *RangeLength* positions. Non-text items (except for the tables⁽¹⁷³⁾ and tabulators⁽¹⁶²⁾) 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`⁽⁹⁸³⁾ unit).

9.5 Functions from RVMathItem Unit

```
procedure RVGetMathProperties(rv: TCustomRichView210;
  var FontName: TFontName; var FontSizeDouble: Integer;
  var TextColor: TColor);
procedure RVSetMathProperties(rv: TCustomRichView210;
  const FontName: String; FontSizeDouble: Integer;
  TextColor: TColor; EditMode: Boolean);
```

RVGetMathProperties returns default values of properties of mathematical expressions⁽¹⁸⁷⁾. If *rv.DocObjects*⁽²²⁹⁾ has an item of *TRVMathDocObject*⁽⁹⁷⁴⁾ class, it returns values of its properties. Otherwise, it returns 'Cambria Math', 20, *c/WindowText*.

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*⁽²²⁹⁾ collection. If such an item does not exist, the procedure creates it. If **EditMode** = *True* and (*rv* is *TCustomRichViewEdit*⁽⁴⁶¹⁾), this assignment is performed as an editing operation (undoable).

```
procedure RVApplyMathProperties(rv: TCustomRichView210);
```

RVApplyMathProperties updates mathematical expressions after changing properties of *TRVMathDocObject*⁽⁹⁷⁴⁾ 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*⁽⁴⁶¹⁾ search options.

Unit *RVMisc*.

Syntax

```
function GetRVESearchOptions(fo: TFindOptions): TRVESearchOptions999;
```

This function always includes *rvseoMultiItem* and *rvseoSmartStart* in the result.

See also

- `TRichViewEdit.SearchText`⁵⁴³.

9.7 GetRVSearchOptions

Converts options of `TFindDialog` to `TRichView`²¹⁰ search options

Unit `RVMisc`.

Syntax

```
function GetRVSearchOptions(fo: TFindOptions): TRVSearchOptions1024;
```

This function always includes `rvsroMultitem` and `rvsroSmartStart` in the result.

See also

- `TRichView.SearchText`³⁴⁶.

9.8 HTML from RTF de-encapsulation

```
unit [VCL/FMX] RVRTFDeEncapsulation / fmxRVRTFDeEncapsulation;
```

type

```
TRVAttachmentPositionEvent = procedure (Pos: Integer;
    var TextToInsert: TRVUnicodeString1032) 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 (RichViewPixelsPerInch¹⁰⁴⁷ is ignored).

It does not take headers⁷⁷⁹ and footers⁷⁷⁸ into account.

```
procedure GetPageSize(RVPrint: TRVPrint759; ARVStyle: TRVStyle630;
  out Width, Height: TRVPixelLength1019);
```

```
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_UnitsToPixels981(RVPrint.Margins768.Left,
    RVPrint.Units770, 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.FixMarginsMode766=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.UnitsPixelsPerInch655, 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: TRVPixelLength1019;

// rvoClientTextWidth must be excluded from RichViewEdit1.Options240.

GetPageSize(RVPrint1, w,h);
RichViewEdit1.MaxTextWidth239 :=
  w - RichViewEdit1.LeftMargin238 - RichViewEdit1.RightMargin244;
RichViewEdit1.MinTextWidth240 := RichViewEdit1.MaxTextWidth239;
RichViewEdit1.Format288;

```

9.10 RVGetFirstEndnote and Others

These functions allow to enumerate endnotes¹⁷⁸, footnotes¹⁸⁰, sidenotes¹⁸¹ and text box items¹⁸³ in a document.

Unit [VCL/FMX] RVNote / fmxRVNote.

```

function RVGetFirstEndnote(RichView: TCustomRichView210;
  AllowHidden: Boolean): TRVEndnoteItemInfo928;
function RVGetNextEndnote(RichView: TCustomRichView210;
  Endnote: TRVEndnoteItemInfo928;
  AllowHidden: Boolean): TRVEndnoteItemInfo928;

function RVGetFirstFootnote(RichView: TCustomRichView210;
  AllowHidden: Boolean): TRVFootnoteItemInfo930;
function RVGetNextFootnote(RichView: TCustomRichView210;
  Footnote: TRVFootnoteItemInfo930;
  AllowHidden: Boolean): TRVFootnoteItemInfo930;

```

(introduced in version 10)

Unit RVSideNote.

```
function RVGetFirstSideNote(RichView: TCustomRichView210;
  AllowHidden: Boolean): TRVSideNoteItemInfo933;
```

```
function RVGetNextSideNote(RichView: TCustomRichView210;
  SideNote: TRVSideNoteItemInfo933;
  AllowHidden: Boolean): TRVSideNoteItemInfo933;
```

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⁽¹⁰¹⁾ notes/text boxes (see *rvprHidden* property⁽¹⁰⁰⁰⁾). To enumerate only visible notes/text boxes, use *rvprShowHiddenText* in **RichView.Options**⁽²⁴⁰⁾ as a value for this parameter.

9.11 RVGetNoteTextStyleNo

Returns index of text style for endnote⁽¹⁷⁸⁾ or footnote⁽¹⁷⁸⁾.

Unit [VCL/FMX] RVNote / fmxRVNote.

```
function RVGetNoteTextStyleNo(RVStyle: TRVStyle630; StyleNo: Integer;
  StyleTemplateName: TRVStyleTemplateName(1027) = ''): Integer;
```

(introduced in version 10, changed in version 14)

This function returns text style (index in **RVStyle.TextStyles**⁽⁶⁵⁴⁾) which can be used for footnote or endnote characters.

If **StyleTemplateName** is not empty, and a style template with this name⁽⁷³⁵⁾ exists in **RVStyle.StyleTemplates**⁽⁶⁵²⁾, 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**⁽⁶⁵⁴⁾[**StyleNo**], but **SubSuperScriptType**⁽⁷⁰⁸⁾ = *rvsssSuperScript*.

In all cases, **Protection**⁽⁷⁰⁵⁾ of the returned text style is [*rvprDoNotAutoSwitch*].

If such text style does not exist, this function adds it to the end of **RVStyle.TextStyles**⁽⁶⁵⁴⁾.

The recommended names of style templates to use with this function:

- 'endnote reference' for endnotes⁽¹⁷⁸⁾ and note references⁽¹⁸⁴⁾ inside endnote text⁽⁹²⁶⁾;
- 'footnote reference' for footnotes⁽¹⁸⁰⁾ and note references⁽¹⁸⁴⁾ inside footnote text⁽⁹²⁶⁾.

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: TRVUnicodeString1032): Boolean;  
function RVIsEmail(const s: TRVUnicodeString1032): Boolean;
```

(introduced in version 10; changed in version 18)

First, RVIsURL calls RVIsCustomURL¹⁰⁵⁸. 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 check¹³², or to find which links are local links when loading RTF and DocX files. Besides, it is used in demo projects and in [RichViewActions](#).

See also: how to create a chat window¹⁰⁷¹.

9.13 RVMathEquationManager

Returns the object containing global properties for math equations¹⁸⁷.

Unit RVCustomMathWrapper.

```
function RVMathEquationManager: TRVCustomMathEquationManager968;
```

(introduced in version 23)

9.14 RVOpenURL

Opens URL in an external browser.

Unit [VCL/FMX] RVFileFuncs / fmxRVFileFuncs.

```
procedure RVOpenURL(const URL: String);
```

(introduced in version 20)

9.15 Unicode Conversion

Various Unicode⁽¹³⁰⁾-related functions.

Unit [VCL / FMX] RVUni / fmxRVUni.

type

```
TRVCodePage = type cardinal; // defined in RVStyle unit
function RVU_AnsiToUnicode(CodePage: TRVCodePage;
  const s: TRVAnsiString(993)): TRVUnicodeString(1032);
function RVU_UnicodeToAnsi(CodePage: TRVCodePage;
  const s: TRVUnicodeString(1032)): TRVAnsiString(993);
function RVU_AnsiToUTF8(CodePage: TRVCodePage;
  const s: TRVAnsiString(993)): TRVRawByteString(1019);
```

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(1019)): 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⁽¹³⁰⁾.

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*⁽⁸⁷⁾.

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"⁽⁸⁷⁾ for details.

10.2 TRVAnsiString

Unit [VCL/FMX] RVTypes / fmxRVTypes;

Declaration for Lazarus⁽¹⁵⁴⁾, 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⁽¹⁵⁴⁾)

See also

- TRVUnicodeString⁽¹⁰³²⁾
- TRVRawByteString⁽¹⁰¹⁹⁾

10.3 TRVBidiMode

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVBidiMode =  
(rvbdUnspecified, rvbdLeftToRight, rvbdRightToLeft);
```

Value of this type defines default text reading direction for document (TRichView.BiDiMode⁽²²⁴⁾), paragraph (TParaInfo.BiDiMode⁽⁷²¹⁾), or text (TFontInfo.BiDiMode⁽⁶⁹⁹⁾).

Value	Meaning
<i>rvbdUnspecified</i>	Bidirectional text is not supported (if specified for document), or text direction is inherited from higher level:

Value	Meaning
	<ul style="list-style-type: none"> ▪ for text – from paragraph; ▪ for paragraph – from document.
<i>rvbdLeftToRight</i>	Default text reading order is left-to-right.
<i>rvbdRightToLeft</i>	Default text reading order is right-to-left.

See also:

Bidirectional text in TRichView ⁽¹³²⁾

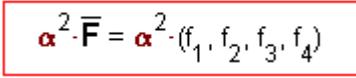
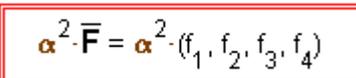
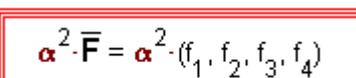
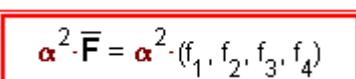
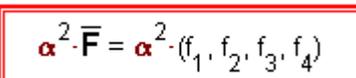
10.4 TRVBorderStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVBorderStyle = (rvbNone, rvbSingle, rvbDouble, rvbTriple,
  rvbThickInside, rvbThickOutside);
```

Style of border around paragraph ⁽⁹⁷⁾ or around a text box (of a sidenote ⁽¹⁸¹⁾ or a text box item ⁽¹⁸³⁾).
Type for Border ⁽⁷²¹⁾.Style ⁽⁷⁴⁴⁾ property.

Style	Border
<i>rvbNone</i>	No border
<i>rvbSingle</i>	
<i>rvbDouble</i>	
<i>rvbTriple</i>	
<i>rvbThickInside</i>	
<i>rvbThickOutside</i>	

(examples are made with Color ⁽⁷⁴⁴⁾=c/Red, Width ⁽⁷⁴⁴⁾=1, InternalWidth ⁽⁷⁴⁴⁾=1)

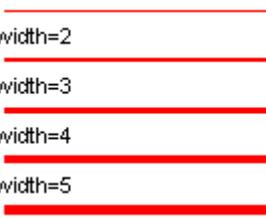
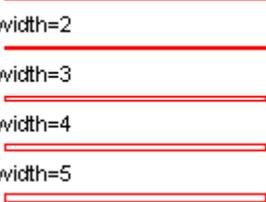
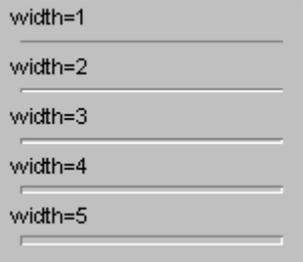
10.5 TRVBreakStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVBreakStyle =
  (rvbsLine, rvbsRectangle,
   rvbs3d, rvbsDotted, rvbsDashed);
```

Style of *break*¹⁶⁷ item.

Style	Example
<i>rvbsLine</i>	<pre>width=1 width=2 width=3 width=4 width=5</pre> 
<i>rvbsRectangle</i>	<pre>width=1 width=2 width=3 width=4 width=5</pre> 
<i>rvbs3d</i>	<pre>width=1 width=2 width=3 width=4 width=5</pre>  <p>(break color is ignored)</p>
<i>rvbsDotted</i>	<pre>width=1 width=2 width=3 width=4 width=5</pre> 

Style	Example
<i>rvbsDashed</i>	<pre>width=1 ----- width=2 ----- width=3 ----- width=4 ----- width=5 -----</pre>

10.6 TRVCellVAlign

Unit [VCL/FMX] RVTable / fmxRVTable;

type

```
TRVCellVAlign =
  (rvcTop, rvcMiddle, rvcBottom, rvcVDefault);
```

Types vertical alignment in cells.

See VAlign⁽⁹¹⁰⁾ property of table rows and VAlign⁽⁹⁰⁶⁾ property of cells.

10.7 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⁽¹⁰⁵⁰⁾) can be used only in several places to specify default code page (which may be different depending on file format or on OS).

10.8 TRVColor

Unit [VCL/FMX] RVTypes / fmxRVTypes;

Color type.

Syntax (VCL and LCL)

type

```
TRVColor = TColor;
```

Syntax (FireMonkey)

type

```
TRVColor = TAlphaColor;
```

See also:

- Color constants⁽¹⁰³⁸⁾

10.9 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
<i>rvcmColor</i>	Color output.
<i>rvcmVCLThemeColor</i> [VCL and LCL]	If a custom function for applying system colors ⁽¹⁰⁵⁷⁾ 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 <i>rvcmColor</i> .
<i>rvcmFMXThemeColor</i> [FMX]	Use a foreground color defined in the component's FireMonkey style instead of black color. Other colors are not affected.
<i>rvcmPrinterColor</i>	Color output, but the following colors are changed: <ul style="list-style-type: none"> ▪ <i>clWindow</i>, <i>clBtnHighlight</i> to <i>clWhite</i>, ▪ <i>clWindowText</i> to <i>clBlack</i>, ▪ <i>clBtnShadow</i> to <i>clGray</i>, ▪ <i>clBtnFace</i>, <i>clScrollbar</i> to <i>clSilver</i>.
<i>rvcmGrayScale</i>	Colors are converted to shades of gray.
<i>rvcmBlackAndWhite</i>	Paragraphs backgrounds are not printed; text is printed black on white or white on black depending on what is darker.
<i>rvcmBlackOnWhite</i>	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* ⁽¹⁶⁷⁾, list markers ⁽¹⁷⁴⁾, tables ⁽¹⁷³⁾.

It does not affect drawing and printing pictures.

10.10 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;
```

10.11 TRVDBFieldFormat

Format for saving data in the database field

Unit [VCL/FMX] RichView / fmxRichView;

type

```
TRVDBFieldFormat =
  (rvdbRVF, rvdbRTF, rvdbDocX, rvdbText, rvdbHTML,
   rvdbMarkdown)
```

Value	Meaning
<i>rvdbRVF</i>	RichView Format (RVF) ⁽¹²⁴⁾ *
<i>rvdbRTF</i>	Rich Text Format (RTF)
<i>rvdbDocX</i>	Microsoft Word Document (DocX)*
<i>rvdbText</i>	Plain text**
<i>rvdbHTML</i>	HTML***
<i>rvdbMarkdown</i>	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` ⁽²³⁷⁾.
`.Encoding` ⁽⁴³¹⁾ and `EncodingStr` ⁽⁴³²⁾.

See also:

- `TRichView` ⁽²¹⁰⁾.`Document` ⁽²³²⁾.`FieldFormat` ⁽⁴²⁴⁾ (live binding)
- `TDBRichViewEdit` ⁽⁶⁰⁶⁾.`FieldFormat` ⁽⁶¹³⁾

10.12 TRVDisplayOptions

Unit [VCL/FMX] RVScroll / fmxRVScroll;

type

```
TRVDisplayOption = (rvdoImages, rvdoComponents, rvdoBullets);
TRVDisplayOptions = set of TRVDisplayOption;
```

const

```
rvdoALL = [rvdoImages, rvdoComponents, rvdoBullets];
```

10.13 TRVDocRotation

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVDocRotation = (rvrotNone, rvrot90, rvrot180, rvrot270);
```

Possible values for a table cell rotation.

Value	Meaning
<i>rvrotNone</i>	Not rotated (0°)
<i>rvrot90</i>	Rotated by 90°
<i>rvrot180</i>	Rotated by 180°
<i>rvrot270</i>	Rotated by 270°

10.14 TRVEnumScope

Unit [VCL/FMX]: RVWordEnum / fmxRVWordEnum;

A word enumeration scope.

Syntax

type

```
TRVEnumScope = (rvesFromCursor, rvesFromStart);
```

Value	Meaning
<i>rvesFromCursor</i>	From the caret position to the end of the document
<i>rvesFromStart</i>	From the beginning to the end of the document

10.15 TRVSearchOptions

Unit [VCL/FMX] RVEdit / fmxRVEdit;

type

```
TRVSearchOption = (rvseoMatchCase, rvseoDown, rvseoWholeWord,
  rvseoMultiItem, rvseoSmartStart, rvseoFromStored);
TRVSearchOptions = set of TRVSearchOption;
```

Options for searching in TRichViewEdit, see TCustomRichViewEdit.SearchText⁵⁴³.

Value	Meaning
<i>rvseoMatchCase</i>	If included, character case is taken into account when comparing string.
<i>rvseoDown</i>	If included, search is performed to the end of document. If not included, it is performed to the top of document.
<i>rvseoWholeWord</i>	If included, the searched string matches only whole words.
<i>rvseoMultitem</i>	If included, the search can match substrings of several text items ¹⁶¹ . If not included, the text is searched in each text item separately. For example, if the document contains the text Hello , the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.
<i>rvseoSmartStart</i>	If included, and a document fragment is selected, then: <ul style="list-style-type: none"> ▪ 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 <i>rvseoFromStored</i> .
<i>rvseoFromStored</i>	If included, searching starts from the position of the last stored search result ³⁶⁷ instead of the current selection or caret position. This option affects only cases when searching starts from a selection.

10.16 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.

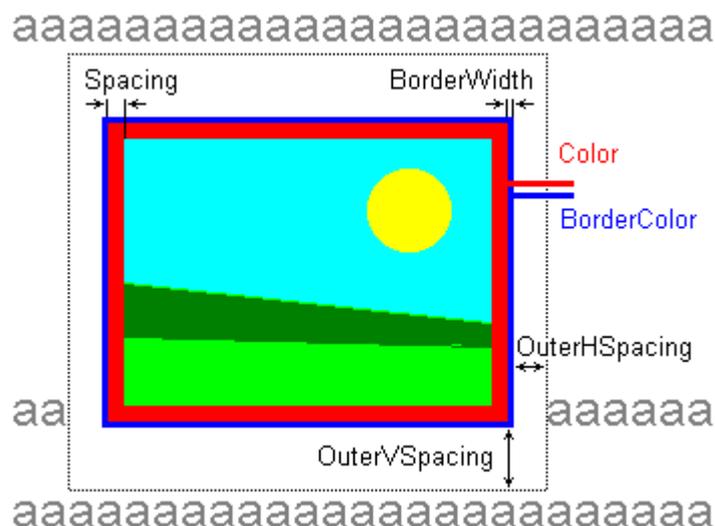
Positioning

Property	Applicable To	Default Value	Meaning
<i>rvepVShift</i>	rect.items*	0	Vertical offset, in % of item height (default), or in RVStyle.Units ⁽⁶⁵⁵⁾ (see <i>rvepVShiftAbs</i>). Negative values move the item down, positive move it up (relative to the position specified in VAlign ⁽¹⁰³³⁾)
<i>rvepVShiftAbs</i>	rect.items*	0	If non-zero, <i>rvepVShift</i> defines vertical offset in RVStyle.Units ⁽⁶⁵⁵⁾ instead of %.
<i>rvepVAlign</i>	rect.items*	ord(<i>rvvaBaseline</i>)	Vertical alignment, TRVVAAlign ⁽¹⁰³³⁾ converted to integer.

* "rect.items" include pictures⁽¹⁶³⁾, hot-pictures⁽¹⁶⁶⁾, controls⁽¹⁶⁸⁾, bullets⁽¹⁷⁰⁾, hotspots⁽¹⁷²⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, note references⁽¹⁸⁴⁾.

Borders and spacing

Scheme:



Property	Applicable To	Default Value	Meaning
<i>rvepSpacing</i>	rect.items*	1	Spacing around the item content, in RVStyle.Units ⁽⁶⁵⁵⁾ . Must be a zero or a positive value. This is a padding, a spacing between the item content and the item border. This area may be colored with <i>rvepColor</i> .
<i>rvepColor</i>	rect.items*	<i>cNone</i>	Background color (TRVColor ⁽⁹⁹⁶⁾ value). It is drawn below the image (and around the image, if <i>rvepSpacing</i> is positive).
<i>rvepBorderWidth</i>	rect.items*	0	Border width, in RVStyle.Units ⁽⁶⁵⁵⁾ . Must be a zero or a positive value.
<i>rvepBorderColor</i>	rect.items*	<i>cBlack</i>	Border color (TRVColor ⁽⁹⁹⁶⁾ value).
<i>rvepOuterHSpacing</i> <i>rvepOuterVSpacing</i>	rect.items*	0	Spacing around the border (horizontal and vertical), in RVStyle.Units ⁽⁶⁵⁵⁾ . Must be a zero or a positive value.

* "rect.items" include pictures⁽¹⁶³⁾, *hot-pictures*⁽¹⁶⁶⁾, controls⁽¹⁶⁸⁾, *bullets*⁽¹⁷⁰⁾, *hotspots*⁽¹⁷²⁾, labels⁽¹⁷⁶⁾, sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, note references⁽¹⁸⁴⁾.

Pictures

Property	Applicable To	Default Value	Meaning
<i>rvepImageWidth</i> , <i>rvepImageHeight</i>	<ul style="list-style-type: none"> ▪ pictures⁽¹⁶³⁾ ▪ <i>hot-pictures</i>⁽¹⁶⁶⁾ 	0	If set to positive value, define size of the image in RVStyle.Units ⁽⁶⁵⁵⁾ . The actual image is not modified, it is just drawn stretched. See also <i>rvepAnimationInterval</i> property below.
[VCL and LCL] <i>rvepTransparent</i> , <i>rvepTransparentMode</i> , <i>rvepTransparentColor</i>	<ul style="list-style-type: none"> ▪ pictures⁽¹⁶³⁾ ▪ <i>hot-pictures</i>⁽¹⁶⁶⁾ (if they are TBitmap)		Correspond to properties of TBitmap: Transparent, TransparentMode, TransparentColor. Limitations: <ul style="list-style-type: none"> ▪ 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

Property	Applicable To	Default Value	Meaning
			colors for 256-color images (probably, it was fixed in last versions of Delphi)
<i>rvepMinHeightOnPage</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ 	0	<p>Affects printing, allows printing large pictures on several pages. Positive value defines minimal height of part of image allowed to be on page.</p> <p>For example:</p> <ul style="list-style-type: none"> ▪ 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. <p>Do not overuse this feature, it has limitations:</p> <ul style="list-style-type: none"> ▪ such images/controls are always printed from new line; ▪ for control, <code>OnPrintComponent</code> ⁽⁸³¹⁾ 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)
<i>rvepNoHTMLImageSize</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ ▪ <i>bullets</i> ⁽¹⁷⁰⁾ ▪ <i>hotspots</i> ⁽¹⁷²⁾ ▪ list markers ⁽¹⁷⁴⁾ 	0	<p>If non-zero, image size is not saved in HTML, even if <i>rvhtmlsiolImageSizes</i> is included in <code>HTMLSaveProperties</code> ⁽²³⁷⁾.<code>ImageOptions</code> ⁽⁴³⁴⁾.</p> <p>This option is ignored for pictures ⁽¹⁶³⁾ and <i>hot-pictures</i> ⁽¹⁶⁶⁾, if <i>rvepImageWidth</i> or <i>rvepImageHeight</i> properties are non-zero.</p>
<i>rvepAnimationInterval</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ 	0	<p>Animation delay in 1/100 of second.</p> <p>If nonzero and <i>rvepImageWidth</i> and/or <i>rvepImageHeight</i> are defined, playing grid animation ⁽¹¹⁹⁾ (in imagewidth x imageheight frame)</p> <p>This feature works only if <code>RVGridAnimate</code> unit is included in the project.</p>
<i>rvepSmoothScaling</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ 	1	<p>If non-zero, <code>TRichView</code> uses smooth resizing method for pictures.</p>

Property	Applicable To	Default Value	Meaning
	<ul style="list-style-type: none"> ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ 		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
<i>rvepResizable</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ ▪ controls ⁽¹⁶⁸⁾ 	1 (for pictures and <i>hot-pictures</i>), 0 (for controls).	<p>Allows resizing with mouse</p> <ul style="list-style-type: none"> ▪ 0 – no resizing; ▪ non-zero value – resizing is allowed.
<i>rvepDeleteProtect</i>	all non-text items	0	<ul style="list-style-type: none"> ▪ 0 – no protection; ▪ non-zero value – item cannot be deleted in editing operations. <p>See also: <i>rvprDeleteProtect</i> option of Protection ⁽⁷⁰⁵⁾ for text style.</p>

Other

Property	Applicable To	Default Value	Meaning
<i>rvepVisible</i>	<ul style="list-style-type: none"> ▪ controls ⁽¹⁶⁸⁾ 	1	Set to 0 to hide the control (it still occupies its place, but is not displayed)
<i>rvepHidden</i>	all non-text items	0	If non-zero, this item is hidden ⁽¹⁰¹⁾ .
<i>rvepShared</i>	<ul style="list-style-type: none"> ▪ pictures ⁽¹⁶³⁾ ▪ <i>hot-pictures</i> ⁽¹⁶⁶⁾ ▪ controls ⁽¹⁶⁸⁾ 	0	If non-zero, the item content (picture or control) is not deleted when the item is deleted.

See also

- TRVExtraItemStrProperty ⁽¹⁰⁰⁵⁾

See also methods of TRichView

- SetItemExtraIntProperty ⁽³⁵⁸⁾,
- GetItemExtraIntProperty ⁽³⁰⁰⁾.

See also methods of TRichViewEdit

- SetItemExtraIntPropertyEd ⁽⁵⁶²⁾,
- SetCurrentItemExtraIntProperty ⁽⁵⁵⁶⁾,

- `GetCurrentItemExtraIntProperty`⁽⁵¹⁰⁾.

10.17 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
<i>rvespUnknown</i>		Not a property. For internal use.
<i>rvespAlt</i>	<ul style="list-style-type: none"> ▪ pictures⁽¹⁶³⁾ ▪ <i>hot-pictures</i>⁽¹⁶⁶⁾ ▪ <i>bullets</i>⁽¹⁷⁰⁾ ▪ <i>hotspots</i>⁽¹⁷²⁾ 	Text representation of image. Saved as "" attribute in HTML files. Must not contain line breaks and double quotes.
<i>rvespHint</i>	all items (but not shown for controls and tables)	<p>Popup hint can be displayed for items if <i>rvoShowItemHints</i> is included in <code>RichView.Options</code>⁽²⁴⁰⁾ and <code>RichView.ShowHint=True</code>.</p> <p>Hints are displayed by assigning to <code>RichView.Hint</code>. See also <code>OnItemHint</code>⁽³⁸¹⁾ event.</p> <p>Note: this feature adds 4 bytes in memory to each item. If you do not need it, enable <code>RVDONOTUSEITEMHINTS</code> compiler define in <code>RV_Defs.inc</code>. After enabling this define, you can still use <code>OnItemHint</code>⁽³⁸¹⁾.</p> <p>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.</p> <p>In <code>OnWriteHyperlink</code>⁽⁴¹¹⁾ event, the proper value of <code>Extras</code> parameter is set by default.</p>
<i>rvespImageFileName</i>	<ul style="list-style-type: none"> ▪ pictures⁽¹⁶³⁾ ▪ <i>hot-pictures</i>⁽¹⁶⁶⁾ ▪ tables⁽¹⁷³⁾ 	Provides a place for storing image file name. For tables, it is a background image, also accessible as <code>table.BackgroundImageFileName</code> ⁽⁸⁵⁰⁾ .

Property	Applicable To	Meaning
		<p>See also: TRVTableCellData.BackgroundImageFileName⁽⁸⁹⁷⁾ <i>rvhtmlsioUseItemImageFileNames</i> is included in HTMLSaveProperties⁽²³⁷⁾.ImageOptions⁽⁴³⁴⁾, 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).</p> <p>In some cases, the component may assign this property, if <i>rvAssignImageFileNames</i> is included in TRichView.Options⁽²⁴⁰⁾.</p>
<i>rvespTag</i>	all items	tag ⁽⁹¹⁾

See also

- TRVExtraItemProperty⁽¹⁰⁰⁰⁾

See also methods of TRichView

- SetItemExtraStrProperty⁽³⁵⁹⁾,
- GetItemExtraStrProperty⁽³⁰⁰⁾.

See also methods of TRichViewEdit

- SetItemExtraStrPropertyEd⁽⁵⁶²⁾,
- SetCurrentItemExtraStrProperty⁽⁵⁵⁷⁾,
- GetCurrentItemExtraStrProperty⁽⁵¹⁰⁾.

10.18 TRVFloatPosition

Unit [VCL/FMX] RVFloatingPos / fmxRVFloatingPos;

type

```
TRVFloatPosition = type Integer;
```

This is a type of HorizontalOffset⁽⁹⁴²⁾ and VerticalOffset⁽⁹⁴⁶⁾ properties of TRVBoxPosition⁽⁹⁴⁰⁾.

Meaning of these values depend on HorizontalPositionKind⁽⁹⁴²⁾ and VerticalPositionKind⁽⁹⁴⁶⁾ properties, and may mean either values in TRVStyleUnits⁽¹⁰²⁷⁾ or in (1/1000)%.

10.19 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⁽⁹⁴⁰⁾:

- HorizontalPositionKind⁽⁹⁴²⁾

- VerticalPositionKind⁹⁴⁶

Value	Meaning
<i>rvfpkAlignment</i>	A position is specified in HorizontalAlignment ⁹⁴² /VerticalAlignment ⁹⁴⁶
<i>rvfpkAbsPosition</i>	HorizontalOffset ⁹⁴² /VerticalOffset ⁹⁴⁶ contains offset in RVStyle.Units ⁶⁵⁵
<i>rvfpkPercentPosition</i>	HorizontalOffset ⁹⁴² /VerticalOffset ⁹⁴⁶ contains offset in (1/1000)% of the anchor area size

10.20 TRVFloatProperty

Unit [VCL/FMX] RichView / fmxRichView;

Identifies a property of TRichView²¹⁰ of a TRVLength¹⁰¹⁵ type.

type

```
TRVFloatProperty = (rvfpDPLeftMargin, rvfpDPRightMargin,
  rvfpDPTopMargin, rvfpDPBottomMargin,
  rvfpDPHeaderY, rvfpDPFooterY);
```

Value	Corresponding property of TRichView
<i>rvfpDPLeftMargin</i>	DocParameters ²³⁰ .LeftMargin ⁴¹⁸
<i>rvfpDPRightMargin</i>	DocParameters.RightMargin ⁴¹⁹
<i>rvfpDPTopMargin</i>	DocParameters.TopMargin ⁴²⁰
<i>rvfpDPBottomMargin</i>	DocParameters.BottomMargin ⁴¹⁶
<i>rvfpDPHeaderY</i>	DocParameters.HeaderY ⁴¹⁸
<i>rvfpDPFooterY</i>	DocParameters.FooterY ⁴¹⁷

A parameter of this type is used in TRichViewEdit⁴⁶¹.SetFloatPropertyEd⁵⁴⁵.

See also:

- TRVIntProperty¹⁰¹³;
- TRVStrProperty¹⁰²⁶.

10.21 TRVFloatSize

Unit [VCL/FMX] RVFloatingBox / fmxRVFloatingBox;

type

```
TRVFloatSize = type Integer;
```

This is a type of Width⁹³⁹ and Height⁹³⁸ properties of TRVBoxProperties⁹³⁷.

Meaning of these values depend on WidthType⁹³⁹ and HeightType⁹³⁸ properties, and may mean either values in TRVStyleUnits¹⁰²⁷ or in (1/1000)%.

10.22 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;
```

10.23 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⁽⁷¹²⁾.

This is a type of the following properties:

- TRVStyleTemplate.ValidTextProperties⁽⁷³⁸⁾;
- TFontInfo.ModifiedProperties⁽⁷¹⁴⁾.

Parameters of this type are used in the following methods:

- TFontInfos.FindSuchStyle, FindSuchStyleEx⁽⁶⁸¹⁾;
- TFontInfo.IsEqual⁽⁷¹⁷⁾.

const

```
RVAllFontInfoProperties: TRVFontInfoProperties =
  [Low(TRVFontInfoProperty)..High(TRVFontInfoProperty)];
```

Constant, lists all TFontInfo properties.

10.24 TRVFontSize

Unit [VCL/FMX]: RVTypes / fmxRVTypes;

Font size type.

Syntax (VCL and LCL)

type

```
TRVFontSize = Integer;
```

Syntax (FireMonkey)

type

```
TRVFontSize = Single;
```

See also:

- TCustomRVFontInfo⁽⁶⁹⁶⁾.Size⁽⁷⁰⁷⁾

10.25 TRVFReaderStyleMode

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVFReaderStyleMode =  
(rvf_sIgnore, rvf_sInsertMap, rvf_sInsertMerge)
```

Defines a mode for loading collection of styles from RVF⁽¹²⁴⁾ (if they are contained there, see *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions⁽²⁴⁷⁾).

This is a type of the following properties of TCustomRichView:

- RVFTextStylesReadMode⁽²⁵¹⁾ (loading Style⁽²⁵³⁾.TextStyles⁽⁶⁵⁴⁾);
- RVFParaStylesReadMode⁽²⁵¹⁾ (loading Style.ParaStyles⁽⁶⁴⁸⁾ and Style.ListStyles⁽⁶⁴⁶⁾).

Mode	When loading RVF...	When Inserting RVF...
<i>rvf_sIgnore</i>	Collections of styles in RVF are ignored. For correct loading, this RVF file must be saved with the same collections of styles.	
<i>rvf_sInsertMerge</i>	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).
<i>rvf_sInsertMap</i>	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⁽⁶³⁰⁾ component is used exclusively by the given TRichView[Edit].

If style templates are used⁽²⁵⁶⁾, and RVFTextStylesReadMode⁽²⁵¹⁾ = RVFParaStylesReadMode⁽²⁵¹⁾ = *rvf_sInsertMerge*, and StyleTemplateInsertMode⁽²⁵⁴⁾ <> *rvstimIgnoreSourceStyleTemplates*, the RVF loading methods replace the existing Style⁽²⁵³⁾.StyleTemplates⁽⁶⁵²⁾ 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`⁽²⁵⁴⁾.

Loading operations include:

- `LoadRVF`⁽³²⁸⁾,
- `LoadRVFFromStream`⁽³²⁷⁾,
- loading from database fields⁽⁵⁹⁴⁾.

Insertion operations include:

- `InsertRVFFromStream`⁽³¹⁶⁾,
- `AppendRVFFromStream`⁽²⁷⁶⁾,
- `TRichViewEdit.InsertRVFFromFileEd`⁽⁵²⁸⁾,
- `TRichViewEdit.InsertRVFFromStreamEd`⁽⁵²⁹⁾,
- pasting⁽¹⁰⁸⁾ from the Clipboard,
- drag&drop⁽¹¹⁸⁾.

10.26 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*
<i>rvgtBitmap</i>	Windows bitmap	All
<i>rvgtIcon</i>	Windows icon	All
<i>rvgtMetafile</i>	Windows metafile, or Windows enhanced metafile	VCL
<i>rvgtJPEG</i>	JPEG image (Joint Photographic Experts Group)	All
<i>rvgtPNG</i>	PNG image (Portable Network Graphics)	All
<i>rvgtGIF</i>	GIF image (Graphics Interchange Format)	VCL, FMX
<i>rvgtTIFF</i>	TIFF image (Tagged Image File Format)	VCL, FMX
<i>rvgtAnyMap</i>	AnyMap image	LCL
<i>rvgtSVG</i>	SVG image (Scalable Vector Graphic)	VCL (requires third-party classes)

Value	Meaning	Supported in frameworks*
		FMX (requires Skia4Delphi)
<i>rvgtWDP</i>	JPEG XR (also known as HD Photo and Windows Media Photo). File extensions: HDP, JXR, WDP.	FMX
<i>rvgtWebP</i>	WebP image	FMX (requires Skia4Delphi)
<i>rvgtOther</i>	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.

10.27 TRVHFType

Type of header or footer

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVHFType = (rvhftNormal, rvhftFirstPage, rvhftEvenPages);
```

Value	Meaning
<i>rvhftNormal</i>	Header/footer for all pages (except for pages having a different header/footer specified)
<i>rvhftFirstPage</i>	Header/footer for the first page
<i>rvhftEvenPages</i>	Header/footer for even pages

10.28 TRVHTML-Encoding

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVHTML-Encoding =  
(rvhtmlUTF8, rvhtmlANSI);
```

Encoding for HTML saving

Value	Meaning
<i>rvhtmlUTF8</i>	UTF-8

Value	Meaning
<i>rvhtmlANSI</i>	ANSI (Windows) encoding, corresponding to TRVStyle ⁽⁶³⁰⁾ .TextStyles ⁽⁶⁵⁴⁾ [0].Charset ⁽⁷⁰⁰⁾

10.29 TRVHTMLLength

Unit [VCL/FMX]: RVTable / fmxRVTable;

type

```
TRVHTMLLength = type Integer;
```

Represents length either in percent or in TRVStyleUnits⁽¹⁰²⁷⁾.

Range	Meaning
>0	Length in TRVStyleUnits ⁽¹⁰²⁷⁾
<0	Length in percent (for example, -50 means 50%)
=0	Length is undefined (default)

See also

- TRVTableItemInfo.BestWidth⁽⁸⁵⁰⁾
- TRVTableCellData.BestWidth⁽⁸⁹⁸⁾

10.30 TRVHTMLSavingPart

Unit [VCL/FMX] RVStyle / fmxRVStyle;

Specifies which part of HTML must be saved.

type

```
TRVHTMLSavingPart = (
  rvhtmlspComplete,
  rvhtmlspFirst,
  rvhtmlspMiddle,
  rvhtmlspLast);
```

Value	Meaning
<i>rvhtmlspComplete</i>	Complete HTML
<i>rvhtmlspFirst</i>	Opening part of HTML (including opening tag of <html>, <head>, and opening tag of <body>)
<i>rvhtmlspMiddle</i>	Content inside <body>
<i>rvhtmlspLast</i>	Closing part of HTML (including </body> and </html>)

10.31 TRVIntProperty

Unit [VCL/FMX] RichView / fmxRichView.

Identifies a property of TRichView⁽²¹⁰⁾ 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
<i>rvipLeftMargin</i>	LeftMargin ⁽²³⁸⁾
<i>rvipRightMargin</i>	RightMargin ⁽²⁴⁴⁾
<i>rvipTopMargin</i>	TopMargin ⁽²⁵⁵⁾
<i>rvipBottomMargin</i>	BottomMargin ⁽²²⁵⁾
<i>rvipMaxTextWidth</i>	MaxTextWidth ⁽²³⁹⁾
<i>rvipMinTextWidth</i>	MinTextWidth ⁽²⁴⁰⁾
<i>rvipBackgroundStyle</i>	BackgroundStyle ⁽²²³⁾
<i>rvipColor</i>	Color ⁽²²⁶⁾
<i>rvipBiDiMode</i>	BiDiMode ⁽²²⁴⁾
<i>rvipDPUnits</i>	DocParameters ⁽²³⁰⁾ .Units ⁽⁴²⁰⁾
<i>rvipDPOrientation</i>	DocParameters.Orientation ⁽⁴¹⁸⁾
<i>rvipDPTitlePage</i>	DocParameters.TitlePage ⁽⁴¹⁹⁾
<i>rvipDPFacingPages</i>	DocParameters.FacingPages ⁽⁴¹⁷⁾
<i>rvipDPMirrorMargins</i>	DocParameters.MirrorMargin ⁽⁴¹⁸⁾

A parameter of this type is used in TRichViewEdit⁽⁴⁶¹⁾.SetIntPropertyEd⁽⁵⁴⁵⁾.

See also:

- TRVFloatProperty⁽¹⁰⁰⁷⁾;
- TRVStrProperty⁽¹⁰²⁶⁾.

10.32 TRVItemBackgroundStyle

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVItemBackgroundStyle =
  (rvbsColor, rvbsStretched, rvbsTiled, rvbsCentered);
```

Defines background style (position of background image).

Value	Meaning
<i>rvbsColor</i>	No image.
<i>rvbsStretched</i>	Image stretched to fill the whole area. Not all image types can be stretched.
<i>rvbsTiled</i>	Image is repeated to fill the whole area.
<i>rvbsCentered</i>	Image is located at the center.

See also:

- TRVTableItemInfo.BackgroundColor⁽⁸⁵⁰⁾ (background of table)
- TRVTableCellData.BackgroundColor⁽⁸⁹⁷⁾ (background of cell)
- Smooth image scaling⁽¹²⁰⁾

10.33 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
<i>rvsfaIgnore</i>	Ignore a single occurrence of a word
<i>rvsfaIgnoreAll</i>	Ignore all occurrences of a word
<i>rvsfaChange</i>	Change one occurrence of a word to another word
<i>rvsfaChangeAll</i>	Change all occurrences of a word to another
<i>rvsfaAdd</i>	Add a word to the dictionary.
<i>rvsfaGuessCompletions</i>	Returns possible completions of a word

See also:

- TRVSpellChecker⁽⁷⁸⁴⁾.OnSpellFormAction⁽⁷⁹⁴⁾
- TRVSpellChecker⁽⁷⁸⁴⁾.OnSpellFormAction⁽⁷⁹⁴⁾

10.34 TRVLength

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVLength = type Extended;
```

This is a type of values measured in TRVUnits⁽¹⁰³²⁾.

10.35 TRVLoadFormat

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```
TRVLoadFormat =  
  (rvlfText, rvlfHTML, rvlfRTF, rvlfRVF,  
   rvlfDocX, rvlfMarkdown, rvlfURL, rvlfOther);
```

Identifies a loading format.

See also:

- TRichView.OnReadHyperlink⁽³⁸⁸⁾

10.36 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
<i>rvdeNone</i>	Drag&drop is not possible
<i>rvdeCopy</i>	The source object will be copied to the target
<i>rvdeMove</i>	The source object will be moved to the target
<i>rvdeLink</i>	Link to the source object will be inserted in the target

See events of TCustomRichViewEdit:

- OnOleDragEnter⁽⁵⁷⁹⁾;
- OnOleDragOver⁽⁵⁸⁰⁾;
- OnOleDrop⁽⁵⁸¹⁾.

10.37 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.

10.38 TRVPageNumberType

Unit [VCL/FMX]: RVFieldItems / fmxRVFieldItems.

type

```
TRVPageNumberType = (rvpntDefault, rvpntDecimal,
    rvpntDecimalDash,
    rvpntLowerAlpha, rvpntUpperAlpha,
    rvpntLowerRoman, rvpntUpperRoman);
```

Numbering type for "page number" field¹⁸⁵.

Type	Meaning
<i>rvpntDefault</i>	Default numbering. Displayed as decimal.
<i>rvpntDecimal</i>	1, 2, 3,...
<i>rvpntDecimalDash</i>	- 1 -, - 2 -, - 3 -
<i>rvpntLowerAlpha</i>	a, b, c, ..., z, aa, ab, ...
<i>rvpntUpperAlpha</i>	A, B, C, ..., Z, AA, AB, ...
<i>rvpntLowerRoman</i>	i, ii, iii, iv, ...
<i>rvpntUpperRoman</i>	I, II, III, IV, ...

This is a type of the following properties:

- TRVPageNumberType.NumberType⁹⁵².

10.39 TRVPageSet

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVPageSet = (rvpsAll, rvpsOdd, rvpsEven);
```

Set of pages.

Value	Meaning
<i>rvpsAll</i>	All pages

Value	Meaning
<i>rvpsOdd</i>	Odd pages
<i>rvpsEven</i>	Even pages

See also:

- TRVPrint⁽⁷⁵⁹⁾.Print⁽⁷⁷⁶⁾
- TRVPrint.PrintPages⁽⁷⁷⁷⁾

10.40 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⁽⁷¹⁸⁾.

This is a type of the following properties:

- TRVStyleTemplate.ValidParaProperties⁽⁷³⁸⁾;
- TParaInfo.ModifiedProperties⁽⁷²⁹⁾.

Parameters of this type are used in the following methods:

- TParaInfos.FindSuchStyle, FindSuchStyleEx⁽⁶⁸⁵⁾;
- TParaInfo.IsEqual⁽⁷³¹⁾.

const

```
RVAllParaInfoProperties: TRVParaInfoProperties =
  [Low(TRVParaInfoProperty)..High(TRVParaInfoProperty)];
```

A constant, lists all TParaInfo properties.

10.41 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: TRVUnicodeString1032;
  Creation, Modified: TDateTime;
  RasterDPI: Single;
  PDFA: Boolean;
  EncodingQuality: Integer;
end;
PRVPDFMetadata = ^TRVPDFMetadata;
```

10.42 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 constants¹⁰⁴²

10.43 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 TRichView¹³⁹

10.44 TRVPixelLength

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVPixelLength = type Integer;
```

This is a type of values measured in pixels, where 1 pixel = 1/TRVStyle⁽⁶³⁰⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾ of an inch.

See also:

Units of measurement in TRichView⁽¹³⁹⁾

10.45 TRVPrintingStep

Unit [VCL/FMX]: PtbIRV / fmxPtbIRV.

type

```
TRVPrintingStep = (rvpsStarting, rvpsProceeding, rvpsFinished);
```

Parameter of this type is used in OnSendingToPrinter⁽⁷⁸²⁾ and OnFormatting⁽⁷⁸⁰⁾ events.

10.46 TRVRawByteString

Unit [VCL/FMX]: RVTypes / fmxRVTypes.

Declaration for Lazarus⁽¹⁵⁴⁾, 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⁽⁹⁹³⁾
- TRVUnicodeString⁽¹⁰³²⁾

10.47 TRVReaderStyleMode

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVReaderStyleMode =  
(rvrsUseSpecified, rvrsUseClosest, rvrsAddIfNeeded);
```

Defines modes for loading formatting from RTF.

Mode	Meaning
<i>rvrsUseSpecified</i>	All character/paragraph formatting in RTF will be ignored, all text/paragraphs will be formatted with the specified style.

Mode	Meaning
<i>rvrsUseClosest</i>	RTF formatting will be mapped to the the most similar existing style. New styles will not be added.
<i>rvrsAddIfNeeded</i>	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` ⁽⁴⁵⁸⁾ and `ParaStyleMode` ⁽⁴⁵⁶⁾.

10.48 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` ⁽³⁹⁷⁾ and `OnSaveItemToFile` ⁽⁴⁰⁴⁾, and `SavePicture` ⁽³⁴²⁾ method.

10.49 TRVSaveOptions

Deprecated. Use `TRichView.HTMLSaveProperties` ⁽²³⁷⁾.

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` ⁽³³⁶⁾, `SaveHTMLEx` ⁽³³⁷⁾, `SaveHTMLToStream` ⁽³⁴⁰⁾, `SaveHTMLToStreamEx` ⁽³⁴⁰⁾.

Options for saving images, both for SaveHTML⁽³³⁶⁾, SaveHTMLEx⁽³³⁷⁾

Option	Meaning
<i>rvsoOverrideImages</i>	If set, TRichView can override existing images. If not set, TRichView will generate unique names for images.
<i>rvsoNoHypertextImageBorders</i>	Obsolete, does nothing
<i>rvsoImageSizes</i>	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 <i>rveplImageWidth</i> and/or <i>rveplImageHeight</i> item extra integer properties ⁽¹⁰⁰⁰⁾ . Sizes are always saved for such images. You can override this setting for the specific item using <i>rveoNoHTMLImagesSize</i> item extra integer properties ⁽¹⁰⁰⁰⁾ .
<i>rvsoUseItemImageFileNames</i>	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 ⁽³⁷⁵⁾ and OnSaveImage2 ⁽⁴⁰²⁾ events have initial value of the Location parameter equal to the image file name. File names for items are defined via <i>rvesplImageFileName</i> string property ⁽¹⁰⁰⁵⁾ .
<i>rvsoInlinelImages</i>	If included, images are saved directly in HTML (in base64 encoding) instead of external files.

Options for saving list markers, both for SaveHTML⁽³³⁶⁾, SaveHTMLEx⁽³³⁷⁾

Option	Meaning
<i>rvsoMarkersAsText</i>	If included, paragraph lists ⁽¹⁷⁴⁾ 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⁽³³⁷⁾); CSS of lists are inserted directly in HTML:

- *rvlstBullet*, *rvlstUnicodeBullet*⁽⁷⁵⁵⁾ – exported as bullets with default markers (disc, circle or square). FormatString⁽⁷⁵²⁾ and Font⁽⁷⁵¹⁾ are ignored;
- *rvlstDecimal*, *rvlstLowerAlpha*, *rvlstUpperAlpha*, *rvlstLowerRoman*, *rvlstUpperRoman*⁽⁷⁵⁵⁾ – exported as lists with the proper numbering. FormatString⁽⁷⁵²⁾ and Font⁽⁷⁵¹⁾ are ignored;
- *rvlstPicture*, *rvlstImageList*, *rvlstImageListCounter*⁽⁷⁵⁵⁾ – exported as a bullet with picture. It's possible to change default image saving by OnHTMLSaveImage⁽³⁷⁵⁾;
- indents are saved, but not very accurately.

Export without CSS (SaveHTML⁽³³⁶⁾):

- *rvlstBullet*, *rvlstUnicodeBullet*, *rvlstPicture*, *rvlstImageList*⁽⁷⁵⁵⁾ – exported as bullets with default markers (disc, circle or square). FormatString⁽⁷⁵²⁾, Picture⁽⁷⁵⁹⁾, ImageList⁽⁷⁵⁴⁾ and Font⁽⁷⁵¹⁾ are ignored;
- *rvlstDecimal*, *rvlstLowerAlpha*, *rvlstUpperAlpha*, *rvlstLowerRoman*, *rvlstUpperRoman*⁽⁷⁵⁵⁾ – exported as lists with the proper numbering. FormatString⁽⁷⁵²⁾ and Font⁽⁷⁵¹⁾ are ignored;
- *rvlstImageListCounter*⁽⁷⁵⁵⁾ – saved as a decimal numbering;
- indents are not saved.

If *rvsoMarkersAsText* is included

Markers are saved without special HTML keywords for lists (like , ,).

Font⁽⁷⁵¹⁾ and format strings⁽⁷⁵²⁾ settings are respected.

Saving without CSS (SaveHTML⁽³³⁶⁾): all indents are ignored.

Saving with CSS (SaveHTMLEx⁽³³⁷⁾): LeftIndent⁽⁷⁵⁵⁾ and MarkerIndent⁽⁷⁵⁷⁾ are retained. FirstIndent⁽⁷⁵¹⁾ 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⁽³³⁶⁾, SaveHTMLEx⁽³³⁷⁾

Option	Meaning
<i>rvsoUseCheckpointsNames</i>	<i>Checkpoints</i> ⁽⁸⁷⁾ names are used instead of checkpoint indices. Recommended.
<i>rvsoXHTML</i>	If included, XHTML is saved instead of HTML
<i>rvsoUTF8</i>	If included, Unicode (UTF-8) HTML file is saved. Highly recommended for multilingual documents.
<i>rvsoDefault0Style</i>	If set, no formatting will be saved for text of the 0-th style (TextStyles ⁽⁶⁵⁴⁾ [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.
<i>rvsoHeaderFooter</i>	If included, a header is saved at the beginning of HTML document, and a footer is saved at the end.

Option	Meaning
	<p>Headers and footers are defined by <code>SetHeader</code>⁽³⁵⁶⁾/<code>SetFooter</code>⁽³⁵⁵⁾ methods.</p> <p>If <code>DocParameters</code>⁽²³⁰⁾.<code>TitlePage</code>⁽⁴¹⁹⁾ = <code>True</code>, the component uses a header and a footer for the first page. Otherwise, a normal header and footer are used.</p> <p>For <code>SaveHTMLEx</code>, headers and footers are always use inline CSS (as if <code>rvsoInlineCSS</code> is included).</p>

Options for SaveHTML⁽³³⁶⁾

Option	Meaning
<code>rvsoForceNonTextCSS</code>	<p>CSS is used for exporting non-text items. This option allows to retain colors of breaks and layout of tables saved with <code>SaveHTML</code> (without using CSS for text and paragraph styles).</p> <p>Note: <code>SaveHTMLEx</code> always uses CSS in this case.</p>

Options for SaveHTMLEx⁽³³⁷⁾

Option	Meaning
<code>rvsoNoDefCSSStyle</code>	<p>By default, <code>SaveHTMLEx</code> assigns attributes of <code>TextStyles</code>⁽⁶⁵⁴⁾[0] to <code><body></code> and <code><table></code>, and attributes of all other styles are saved relative to attributes of <code>TextStyles</code>[0].</p> <p>If you include this option, <code>TextStyles</code>[0] will be treated like any other style. The same for <code>ParaStyles</code>⁽⁶⁴⁸⁾[0].</p> <p>It's not recommended to use this option: it generates much larger CSS and HTML.</p> <p>This option is useful if HTML generated by <code>TRichView</code> is used as a part of larger HTML, and you do not want to change properties of <code><body></code> and <code><table></code>.</p>
<code>rvsoInlineCSS</code>	<p>If included, CSS is inserted directly in <code><p></code>, <code></code> and <code></code> tags (instead of generating CSS table at the beginning of HTML).</p> <p>This option generates highly overbloated HTML. Not recommended.</p>

Options for saving only part of document, both for SaveHTML³³⁶, SaveHTMLEx³³⁷

These options are useful when generating HTML documents what will be inserted in larger HTML document.

Option	Meaning
<i>rvsoFirstOnly</i>	If set, TRichView saves only opening part of HTML (opens <html> tag, saves full <head> tag, opens <body> tag)
<i>rvsoMiddleOnly</i>	If set, TRichView saves only body of HTML document (content inside <body> tag)
<i>rvsoLastOnly</i>	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).

10.50 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 TRVStyle⁶³⁰.UnitsPixelsPerInch⁶⁵⁵ (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 TRichView (see DocumentPixelsPerInch²³³).

This structure is filled automatically, you do not need to use its members.

10.51 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.SearchText³⁴⁶.

Value	Meaning
<i>rvsroMatchCase</i>	If included, a character case is taken into account when comparing strings.
<i>rvsroDown</i>	If included, the search is performed to the end of the document. If not included, it is performed to the top of the document.
<i>rvsroWholeWord</i>	If included, the searched string matches only whole words.
<i>rvsroFromStart</i>	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 <i>rvsroFromStored</i> .
<i>rvsroMultitem</i>	If included, the search can match substrings of several text items ¹⁶¹ . If not included, the text is searched in each text item separately. For example, if the document contains the text Hello , the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.
<i>rvsroSmartStart</i>	This option is used only when <i>rvsroFromStart</i> is not included. If included, and a document fragment is selected, then: <ul style="list-style-type: none"> ▪ 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 <i>rvsroFromStored</i> .
<i>rvsroFromStored</i>	If included, searching starts from the position of the last stored search result ³⁶⁷ 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 <i>rvsroFromStart</i> option).

10.52 TRVSeqType

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVSeqType = (rvseqDecimal, rvseqLowerAlpha,
              rvseqUpperAlpha, rvseqLowerRoman, rvseqUpperRoman);
```

Numbering type for sequences ⁽¹⁷⁷⁾, endnotes ⁽¹⁷⁸⁾, footnotes ⁽¹⁸⁰⁾, sidenotes ⁽¹⁸¹⁾.

Type	Meaning
<i>rvseqDecimal</i>	1, 2, 3,...
<i>rvseqLowerAlpha</i>	a, b, c, ..., z, aa, ab, ...
<i>rvseqUpperAlpha</i>	A, B, C, ..., Z, AA, AB, ...
<i>rvseqLowerRoman</i>	i, ii, iii, iv, ...
<i>rvseqUpperRoman</i>	I, II, III, IV, ...

This is a type of the following properties:

- TRVStyle.EndnoteNumbering ⁽⁶³⁷⁾;
- TRVStyle.FootnoteNumbering ⁽⁶³⁷⁾;
- TRVStyle.SidenoteNumbering ⁽⁶³⁷⁾;
- TRVSeqItemInfo.NumberType ⁽⁹²³⁾.

10.53 TRVStrProperty

Unit [VCL/FMX]: RichView / fmxRichView.

Identifies a property of TRichView ⁽²¹⁰⁾ of a string type.

type

```
TRVStrProperty = (rvspDPTitle, rvspDPAuthor, rvspDPComments);
```

Value	Corresponding property of TRichView
<i>rvspDPTitle</i>	DocParameters ⁽²³⁰⁾ .Title ⁽⁴¹⁹⁾
<i>rvspDPAuthor</i>	DocParameters.Author ⁽⁴¹⁶⁾
<i>rvspDPComments</i>	DocParameters.Comments ⁽⁴¹⁷⁾

A parameter of this type is used in TRichViewEdit ⁽⁴⁶¹⁾.SetStrPropertyEd ⁽⁵⁴⁵⁾.

See also:

- TRVFloatProperty ⁽¹⁰⁰⁷⁾;
- TRVIntProperty ⁽¹⁰¹³⁾.

10.54 TRVStyleLength

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

type

```
TRVStyleLength = type Integer;
```

This is a type of values measured in TRVStyleUnits⁽¹⁰²⁷⁾ (i.e. pixels, twips or EMU).

See also:

Units of measurement in TRichView⁽¹³⁹⁾

10.55 TRVStyleTemplateId

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVStyleTemplateId = type Integer;
```

Values of this type identify items in TRVStyleTemplateCollection⁽⁶⁸⁸⁾,

This is a type of the following properties:

- Id⁽⁷³⁴⁾ of TRVStyleTemplate⁽⁷³³⁾ (unique style template identifier);
- ParentId⁽⁷³⁷⁾ of TRVStyleTemplate⁽⁷³³⁾ (a reference to the parent style template);
- NextId⁽⁷³⁶⁾ of TRVStyleTemplate⁽⁷³³⁾ (a reference to the style template for following paragraphs);
- StyleTemplateId⁽⁶⁹⁶⁾ of text and paragraph attributes (styles), linking them to a style template.
- ParaStyleTemplateId⁽⁷⁰⁵⁾ 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.

10.56 TRVStyleTemplateName

Unit [VCL/FMX]: RVStyle / fmxRVStyle.

type

```
TRVStyleTemplateName = type TRVUnicodeString(1032);
```

(changed in version 18)

This is a type of the following properties:

- Name of TRVStyleTemplate⁽⁷³³⁾ (unique style template name)

10.57 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
<i>rvstuPixels</i>	Pixels.

Value	Meaning
	1 pixel = 1/TRVStyle ⁽⁶³⁰⁾ .UnitsPixelsPerInch ⁽⁶⁵⁵⁾ of an inch (normally, 1/96 of an inch)
<i>rvstuTwips</i>	Twips. 1 twip = 1/20 of a point = 1/1440 of an inch
<i>rvstuEMU</i>	English metric units. 1 EMU = 1/914400 of an inch = 1/36000 mm

Values measured in TRVUnits have type TRVStyleLength⁽¹⁰²⁷⁾.

Pixels

This is a type of TRVStyle⁽⁶³⁰⁾.Units⁽⁶⁵⁵⁾. If it is equal to *rvstuPixels*, values are measured in pixels with DPI specified in TRVStyle.UnitsPixelsPerInch⁽⁶⁵⁵⁾; when rendering on a device (screen or printer), lengths are recalculated according to the device DPI:

$$\langle \text{length on device} \rangle = \langle \text{value measured in TRVStyleUnits} \rangle * \langle \text{device DPI} \rangle \text{ div UnitsPixelsPerInch.}$$

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 ⁽¹³⁹⁾

10.58 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 ⁽⁸⁵²⁾ and CellBorderStyle ⁽⁸⁵⁴⁾ properties of table.

Value	Meaning
<i>rvtbRaised</i>	Raised 3d border, with default colors.
<i>rvtbLowered</i>	Sunken 3d border, with default colors.
<i>rvtbColor</i>	Flat border of BorderColor ⁽⁸⁵¹⁾ /CellBorderColor ⁽⁸⁵³⁾ color.
<i>rvtbRaisedColor</i>	Raised 3d border: <ul style="list-style-type: none"> ▪ left and top sides: BorderLightColor ⁽⁸⁵²⁾ /CellBorderLightColor ⁽⁸⁵⁴⁾; ▪ bottom and right sides: BorderColor ⁽⁸⁵¹⁾ /CellBorderColor ⁽⁸⁵³⁾).
<i>rvtbLoweredColor</i>	Sunken 3d border: <ul style="list-style-type: none"> ▪ left and top sides: BorderColor ⁽⁸⁵¹⁾ /CellBorderColor ⁽⁸⁵³⁾); ▪ bottom and right sides: BorderLightColor ⁽⁸⁵²⁾ /CellBorderLightColor ⁽⁸⁵⁴⁾).

See also

- Colors and layout of tables ⁽¹⁹³⁾

10.59 TRVTag

Unit [VCL/FMX] RVStyle / fmxRVStyle.

type

```
TRVTag = type TRVUnicodeString (1032);
```

This is the type of item tags – additional value associated with any item ⁽¹⁵⁸⁾ in TRichView ⁽²¹⁰⁾ document, and with any table cell ⁽⁹⁰⁶⁾.

See also:

- Tags ⁽⁹¹⁾

10.60 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.

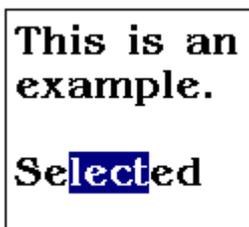
See:

- TRVStyle.OnDrawStyleText⁽⁶⁷¹⁾, OnDrawTextBack⁽⁶⁷³⁾;
- TFontInfo.Draw⁽⁷¹⁶⁾

State	Meaning
<i>rvtsSelected</i>	text is selected (for copying to the Clipboard)
<i>rvtsHover</i>	[hyper]text is below a mouse pointer
<i>rvtsItemStart</i>	this call of event draws text item from the beginning
<i>rvtsItemEnd</i>	this call of event finishes drawing of text item
<i>rvtsDrawItemStart*</i>	this call of event draws part of text item (called "drawing item")from the beginning
<i>rvtsDrawItemEnd*</i>	this call of event finishes drawing of part of text item.
<i>rvtsControlFocused</i>	TRichView has input focus
<i>rvtsSpecialCharacters</i>	<i>rvoShowSpecialCharacters</i> is included in TRichView.Options ⁽²⁴⁰⁾ .

* 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`⁶⁷¹ 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 call was 'lect' with `[rvtsSelected]`).

10.61 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`⁹¹) 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
<code>rvutNone</code>	No undo/redo available (disable an undo or a redo menu item or button)
<code>rvutCustom</code>	The undo action has a special name (see <code>BeginUndoCustomGroup</code> ⁴⁹⁶).

10.62 TRVUnicodeString

Unit [VCL/FMX]: RVTypes / fmxRVTypes

Declaration for Lazarus⁽¹⁵⁴⁾, 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⁽⁹⁹³⁾
- TRVRawByteString⁽¹⁰¹⁹⁾

10.63 TRVUnits

Unit [VCL/FMX] RVStyle / fmxRVStyle.

Units of measurement.

type

```
TRVUnits = (rvuInches, rvuCentimeters, rvuMillimeters, rvuPicas,  
            rvuPixels, rvuPoints);
```

Value	Meaning
<i>rvuInches</i>	Inches. 1 inch = 2.54 cm.
<i>rvuCentimeters</i>	Centimeters.
<i>rvuMillimeters</i>	Millimeters.
<i>rvuPicas</i>	Picas (used in typography). 1 pica = 1/6 of an inch = 12 points.
<i>rvuPixels</i>	Pixels. 1 pixel = 1/TRVStyle ⁽⁶³⁰⁾ .UnitsPerInch ⁽⁶⁵⁵⁾ of an inch. (normally, 1/96 of an inch)
<i>rvuPoints</i>	Points (used in typography). 1 point = 1/72 of an inch.

This is a type of the following properties:

- TRichView⁽²¹⁰⁾.DocParameters⁽²³⁰⁾.Units⁽⁴²⁰⁾.
- TRVPrint⁽⁷⁵⁹⁾.Units⁽⁷⁷⁰⁾

Values measured in TRVUnits have the type TRVLength⁽¹⁰¹⁵⁾.

See also:Units of measurement in TRichView ¹³⁹

10.64 TRVVAAlign

Unit [VCL/FMX] RVStyle / fmxRVStyle;

type

```

TRVVAAlign = (
  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
<i>rvvaBaseline</i>	<u>Text String</u> 
<i>rvvaMiddle</i>	<u>Text String</u> 
<i>rvvaAbsTop</i>	<u>Text string</u> 
<i>rvvaAbsBottom</i>	<u>Text string</u> 
<i>rvvaAbsMiddle</i>	<u>Text String</u> 

Align	Scheme
<i>rvvaLeft</i>	<pre> text text text text text text text text [] text </pre>
<i>rvvaRight</i>	<pre> text [] text </pre>

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 ⁽¹⁶³⁾ and *hot-pictures* ⁽¹⁶⁶⁾
- *bullets* ⁽¹⁷⁰⁾ and *hotspots* ⁽¹⁷²⁾
- controls ⁽¹⁶⁸⁾
- label items ⁽¹⁷⁶⁾
- numbered sequences ⁽¹⁷⁷⁾

See also methods of TCustomRichView ⁽²¹⁰⁾:

- GetItemVAlign ⁽³⁰³⁾,
- SetItemVAlign ⁽³⁶¹⁾.

See also properties of TCustomRichView ⁽²¹⁰⁾:

- ClearLeft ⁽²²⁵⁾,
- ClearRight ⁽²²⁶⁾.

See also methods of TCustomRichViewEdit ⁽⁴⁶¹⁾:

- GetCurrentItemVAlign ⁽⁵¹²⁾,
- SetItemVAlignEd ⁽⁵⁶⁵⁾,
- SetCurrentItemVAlign ⁽⁵⁶⁵⁾,
- ClearTextFlow ⁽⁵⁰¹⁾.

See also properties of TRVStyle ⁽⁶³⁰⁾:

- FloatingLineColor ⁽⁶³⁸⁾

10.65 TRVYesNoAuto

Unit [VCL/FMX] RichView / fmxRichView;

Yes/No/Auto values

Syntax:

type

```
TRVYesNoAuto = (rvynaNo, rvynaYes, rvynaAuto);
```

10.66 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⁽¹⁰⁴⁰⁾ define the caret width;
- RichViewEditCaretHeightExtra⁽¹⁰⁴⁰⁾: Integer, allows making the caret shorter or longer;
- RichViewEditMinCaretHeight, RichViewEditMaxCaretHeight⁽¹⁰⁴⁰⁾: Integer – minimum/maximum possible caret height;
- RichViewEditCustomCaretSize⁽¹⁰⁴⁰⁾: Boolean allows changing caret size in an event.
- RichViewEditCaretPosition⁽¹⁰⁴⁰⁾: TRVHorizontalCaretPosition defines how the caret is aligned horizontally relative to the insertion point.

Editing, Other

- RichViewEditDefaultProportionalResize⁽¹⁰⁴⁵⁾: Boolean, specifies whether images are resized proportionally with the mouse;
- RichViewEditEnterAllowsEmptyMarkeredLines⁽¹⁰⁴⁶⁾: Boolean, enables or disables autodeletion of list marker from new empty lines;
- RichViewUnicodeInput⁽¹⁰⁵⁰⁾: TRichViewUnicodeInput, specifies how types Unicode characters are processed (in non-Unicode versions of Delphi);
- RichViewTableAutoAddRow⁽¹⁰⁴⁹⁾: Boolean, allows or disallows adding new table rows using Tab key, as well as deletion of table rows and columns using Backspace key.
- RichViewCtrlUpDownScroll⁽¹⁰⁴⁴⁾: Boolean, specifies how Ctrl + Up and Ctr + Down keys are processed.

Saving and Loading

- RVDefaultLoadProperties⁽¹⁰⁵¹⁾: TRVDefaultLoadProperties⁽⁹⁶³⁾, default properties for text and paragraph loaded from Markdown, HTML, RTF, DocX.
- RichViewSavePinHTML⁽¹⁰⁴⁰⁾: Boolean, instructs always using <p> for saving in HTML;
- RichViewSaveDivInHTMLEx⁽¹⁰⁴⁰⁾: Boolean, instructs always using <div> for saving in HTML;
- RichViewCSSUseSystemColors⁽¹⁰⁴⁴⁾: Boolean enables/disables saving system colors in CSS;
- RichViewSavePageBreaksInText⁽¹⁰⁴⁰⁾: Boolean, allows or disallows saving page breaking characters in plain text;
- RichViewDoNotCheckRVFStyleRefs⁽¹⁰⁴⁵⁾: Boolean, allows turning off checking for invalid style indices when loading RVF;

- RichViewResetStandardFlag⁽¹⁰⁴⁷⁾: Boolean, specifies whether to reset Standard⁽⁶⁹⁵⁾ property to False when inserting RVF;
- RichViewPixelsPerInch⁽¹⁰⁴⁷⁾: Integer, allows overriding the screen resolution;
- RichViewAutoAssignNormalStyleTemplate⁽¹⁰⁴³⁾: Boolean, specifies whether to assign "Normal" style template when loading unstyled paragraphs from RVF and RTF.
- RichViewTableDefaultRTFAutofit⁽¹⁰⁴⁹⁾: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.
- RichViewSaveHyperlinksInDocXAsFields⁽¹⁰⁴⁸⁾ defines how hyperlinks are saved in DocX.
- DefaultRVFPixelsPerInch⁽¹⁰⁴²⁾ defines "pixels per inch" value for old RVF files.

Tables

- RichViewTableAutoAddRow⁽¹⁰⁴⁹⁾: Boolean, allows or disallows adding new table rows using Tab key, and deleting rows/columns using Backspace key;
- RichViewAllowCopyTableCells⁽¹⁰⁴²⁾: Boolean allows preventing copying multicell selection in a table to the Clipboard.
- RichViewTableDefaultRTFAutofit⁽¹⁰⁴⁹⁾: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.

Formatting and Displaying

- RichViewJustifyBeforeLineBreak⁽¹⁰⁴⁶⁾: Boolean, specifies whether lines before line breaks are justified (expanded);
- RichViewStringFormatMethod⁽¹⁰⁴⁸⁾: Boolean, allows activating an alternative text measuring method;
- RichViewShowGhostSpaces⁽¹⁰⁴⁸⁾: Boolean, allows showing space characters "hidden" between lines.
- RVVisibleSpecialCharacters⁽¹⁰⁶¹⁾: TRVSpecialCharacters specifies which special characters can be shown.
- RVAlwaysShowPlaceholders⁽¹⁰⁵¹⁾: Boolean, allows showing placeholders for text boxes even if special characters are not shown.
- RVThumbnailMaker⁽¹⁰⁶¹⁾: TRVThumbnailMaker⁽⁹⁷⁹⁾, provides access to an object used to create scaled images;
- RichViewMaxPictureCount⁽¹⁰⁴⁶⁾: Integer – maximal count of "active" images in TRichView control.
- RichViewMaxThumbnailCount⁽¹⁰⁴⁷⁾: Integer – maximal count of scaled images stored in TRichView control.

Printing

- RichViewPixelsPerInch⁽¹⁰⁴⁷⁾: Integer, allows overriding the screen resolution;
- RichViewAlternativePicPrint⁽¹⁰⁴³⁾: Boolean, allows using an alternative picture printing method.

Live spelling check

- RichViewApostropheInWord⁽¹⁰⁴³⁾: Boolean, specifies how apostrophe characters are processed when finding word boundaries.

Classes

- RVThumbnailMaker⁽¹⁰⁶¹⁾: TRVThumbnailMaker⁽⁹⁷⁹⁾, provides access to an object used to create scaled images;

- `RVGraphicHandler`¹⁰⁵⁷: `TRVGraphicHandler`⁹⁷², provides access to an object used to manage graphic classes.

Other

- `RichViewCompareStyleNames`¹⁰⁴⁴: Boolean, specifies whether `StyleName`⁶⁹⁶ property is used when comparing (and searching for) text, paragraph or list styles.

Global Functional Variables

- `RVIsCustomURL`¹⁰⁵⁸: `TCustomRVIsURLFunction`, allows defining your own URL prefixes.
- `RVGetSystemColor`¹⁰⁵⁷: `TRVGetSystemColorFunction`, allows using custom color schemes.

Constants

- `rvs***` constants¹⁰⁵⁹: item types;
- `rveipc***` constants¹⁰⁵¹: integer properties of items;
- `rvespc***` constants¹⁰⁵⁶: string properties of items;
- Constants and variables related to Clipboard¹⁰³⁷.
- Color constants¹⁰³⁸
- Pen style constants¹⁰⁴²

11.1 Clipboard Related Constants and Variables

Unit `[VCL/FMX]`: `RVStr / fmxRVStr`;

Clipboard Format Names

Names of the Clipboard formats for copying `RVF`¹²⁴, `RTF`, `URL`, `HTML`.

Syntax

const

```
RVFORMATNAME    = 'RichView Format';
RTFFORMATNAME   = '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(RTFFORMATNAME);
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:

- TRVColor⁹⁹⁶ 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* if you want `SaveHTML`⁽³³⁵⁾ to write `<p>` instead of `<div>` in HTML when `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *rvhtmlstSimplified*. This variable does not affect saving when `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *rvhtmlstNormal*. This setting changes appearance of resulting HTML documents (usually browsers add additional spacing between `<p>`).

Set **RichViewSaveDivInHTMLEx** to *True* if you want `SaveHTML`⁽³³⁵⁾ to write `<div>` instead of `<p>` in HTML when `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *rvhtmlstNormal*. This variable does not affect saving when `HTMLSaveProperties`⁽²³⁷⁾.`HTMLSavingType`⁽⁴³³⁾ = *rvhtmlstSimplified*. This setting does not change appearance of resulting HTML documents (because all `<p>` or `<div>` 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`⁽⁴⁶¹⁾.

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
<i>rvcwmDefault</i>	The caret width is calculated according to RichViewEditCaretWidth , see below.
<i>rvcwmCharacter</i>	The caret has width of the next character/item. When the caret is at the end of line, its width is calculated like in <i>rvcwmDefault</i> 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`⁽⁵⁷⁸⁾ 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
<i>rvhcpDefault</i>	To the left side of the insertion position for items having <code>BiDiMode = rvbdRightToLeft</code> ⁽⁹⁹³⁾ , to the right side otherwise.
<i>rvhcpCenter</i>	Centered at the insertion position
<i>rvhcpRight</i>	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 ¹⁰¹⁸

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 ¹²⁴).

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⁽²²⁸⁾ 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⁽²²⁸⁾ and assign RichViewApostropheInWord=*False*.

Note: apostrophe is not included in Delimiters⁽²²⁸⁾ 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⁽⁶⁹⁶⁾ to any style template⁽⁷³³⁾) are loaded from RTF and RVF files/streams, if style templates are used⁽²⁵⁶⁾.

If *True*, "Normal" style template⁽⁶⁸⁹⁾ (if it exists) is assigned to paragraph styles⁽⁷²⁷⁾ 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⁽²⁵⁴⁾ property.

See also:

- Styles and style templates ⁹⁸.

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` ⁶⁹⁶ 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
<i>clWindow</i>	canvas
<i>clWindowText</i>	canvastext
<i>clBtnFace</i>	buttonface
<i>clBtnText</i>	buttontext
<i>clGrayText</i>	graytext
<i>clHighlight</i>	highlight
<i>clHighlightText</i>	hightlighttext

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 ⁴⁶¹.

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 ⁽¹²⁴⁾.

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 ⁽⁹⁶⁹⁾.

11.15 RichViewDoNotMergeNumbering

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

const

```
RichViewDoNotMergeNumbering: Boolean = False;
```

RichViewDoNotMergeNumbering affects inserting RVF ⁽¹²⁴⁾. 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⁽²¹⁰⁾ in an "active" form, i.e. as TRVGraphic⁽⁹⁷⁰⁾ 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⁽¹²⁰⁾
- RichViewMaxThumbnailCount⁽¹⁰⁴⁷⁾

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⁽²¹⁰⁾.

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⁽¹²⁰⁾
- RichViewMaxPictureCount⁽¹⁰⁴⁶⁾

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⁽⁶³⁰⁾.UnitsPixelsPerInch⁽⁶⁵⁵⁾
- TRichView⁽²¹⁰⁾.DocumentPixelsPerInch⁽²³³⁾

11.22 RichViewResetStandardFlag

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

Syntax

const

```
RichViewResetStandardFlag: Boolean = True;
```

If *True*, Standard⁽⁶⁹⁵⁾ properties of styles which were added from inserted RVF⁽¹²⁴⁾ will be reset to *False* (it only affects RVF insertion, not loading).

If this flag is *True*, new styles having Standard⁽⁶⁹⁵⁾ = *True* may appear only after calling LoadRVF⁽³²⁸⁾ or LoadRVFFromStream⁽³²⁹⁾.

11.23 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⁽³³³⁾ 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.

11.24 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⁽²⁴⁰⁾. Caret cannot be placed after these spaces.

11.25 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
<i>rvsfmAuto</i>	The components tries to choose the fastest method
<i>rvsfmEngine</i>	The component uses the function provided by API
<i>rvsfmTRichView</i>	The component uses its own method (a binary search on string widths).

About the "auto" method

If TextEngine⁽⁶⁵⁴⁾ = *rvteUniscribe*, or in ScaleRichView, the components choose the TRichView method only if the string length is much greater than an estimated count of character fitting to the specified width.

Otherwise, the components always choose the TRichView method.

11.26 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⁽¹⁷³⁾ 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⁽⁴⁶¹⁾ and TDBRichViewEdit⁽⁶⁰⁶⁾.

11.27 RichViewTableDefaultRTFAutoFit

Unit [VCL/FMX]: RVTable / fmxRVTable;

Syntax

const

```
RichViewTableDefaultRTFAutoFit: Boolean = False;
```

Assigning *True* to this variable works like inclusion of *rvtoRTFAutoFit* in Options⁽⁸⁶¹⁾ properties of all tables⁽¹⁷³⁾.

11.28 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⁽¹³⁰⁾ 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.

11.29 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⁽³²⁰⁾ and LoadFromStream, LoadFromStreamEx⁽³²¹⁾;
- TRichView⁽²¹⁰⁾.Document⁽²³²⁾.CodePage⁽⁴²³⁾
- TDBRichView⁽⁵⁹⁴⁾.CodePage⁽⁵⁹⁹⁾
- TDBRichViewEdit⁽⁶⁰⁶⁾.CodePage⁽⁶¹²⁾

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.

11.30 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 ⁽¹⁸³⁾.

If *False*, these icons are shown only at the caret position, or if *rvoShowSpecialCharacters* is included in *TCustomRichView.Options* ⁽²⁴⁰⁾, and *rvscPlaceholders* is included in *RVVisibleSpecialCharacters* ⁽¹⁰⁶¹⁾.

11.31 RVControlsPainter

Unit [VCL/FMX]: RVControls / fmxRVControls;

Syntax

```
var RVControlsPainter: TRVControlsPainter (962);
```

(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*.

11.32 RVDefaultLoadProperties

Unit [VCL/FMX]: RVDefReadProps / fmxRVDefReadProps;

Syntax

var

```
RVDefaultLoadProperties: TRVDefaultLoadProperties (963);
```

(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.

11.33 rveipc*** constants

These constants identify integer properties of *TRichView* items ⁽¹⁵⁸⁾.

They are used in the following methods:

- *TRichView.GetExtraltemIntPropertyEx* ⁽³⁰⁰⁾
- *TRichView.SetExtraltemIntPropertyEx* ⁽³⁵⁸⁾
- *TRichViewEdit.SetItemExtraIntPropertyExEd* ⁽⁵⁶²⁾
- *TRichViewEdit.GetCurrentItemExtraIntPropertyEx* ⁽⁵¹⁰⁾
- *TRichViewEdit.SetCurrentItemExtraIntPropertyEx* ⁽⁵⁵⁶⁾

Properties of Bullets ⁽¹⁷⁰⁾ and Hotspots ⁽¹⁷²⁾

These properties are defined in RVItem unit.

These properties provides an alternative to the following methods:

- TRichView: SetBulletInfo ⁽³⁵²⁾, GetBulletInfo ⁽²⁸⁹⁾, GetHotspotInfo ⁽²⁹⁵⁾, SetHotspotInfo ⁽³⁵⁷⁾;
- TRichViewEdit: SetBulletInfoEd ⁽⁵⁴⁸⁾, SetCurrentBulletInfo ⁽⁵⁵²⁾, GetCurrentBulletInfo ⁽⁵⁰⁵⁾, SetHotspotInfoEd ⁽⁵⁶⁰⁾, SetCurrentHotspotInfo ⁽⁵⁵⁵⁾, GetCurrentHotspotInfo ⁽⁵⁰⁷⁾.

Constant	Value	Property
<i>rveipcImageIndex</i>	200	Image index (for <i>bullets</i> and <i>hotspots</i>)
<i>rveipcHotImageIndex</i>	201	Image index below the mouse pointer (for <i>hotspots</i>)

Properties of Controls ⁽¹⁶⁸⁾

These properties are defined in RVItem unit.

Constant	Value	Property
<i>rveipcPercentWidth</i>	200	If the value is in the range 1..100, it defines the width of the control in % of document width.
<i>rveipcDPIScalable</i>	201	If non-zero (default), controls are scaled from RVF's PPI (pixels per inch) to the actual PPI. See also: RichViewPixelsPerInch ⁽¹⁰⁴⁷⁾

Properties of Breaks ⁽¹⁶⁷⁾

These properties are defined in RVItem unit.

These properties provide an alternative to the following methods:

- TRichView: SetBreakInfo ⁽³⁵¹⁾, GetBreakInfo ⁽²⁸⁹⁾;
- TRichViewEdit: SetBreakInfoEd ⁽⁵⁴⁷⁾, SetCurrentBreakInfo ⁽⁵⁵¹⁾, GetCurrentBreakInfo ⁽⁵⁰⁴⁾.

Constant	Value	Property
<i>rveipcBreakWidth</i>	200	Line width
<i>rveipcBreakStyle</i>	201	Line style*
<i>rveipcBreakColor</i>	202	Line color

* the property is type-casted to Integer

Properties of Label items ⁽¹⁷⁶⁾

These properties are defined in RVLabelItem unit.

They identify properties of label items ⁽¹⁷⁶⁾ and items of all types inherited from them (numbered sequences ⁽¹⁷⁷⁾, footnotes ⁽¹⁸⁰⁾, endnotes ⁽¹⁷⁸⁾, references to notes ⁽¹⁸⁴⁾).

Constant	Value	Property
<i>rveipcProtectTextStyleNo</i>	200	ProtectTextStyleNo ⁽⁹¹⁴⁾ *
<i>rveipcRemoveInternalLeading</i>	201	RemoveInternalLeading ⁽⁹¹⁵⁾ *
<i>rveipcCursor</i>	202	Cursor ⁽⁹¹⁴⁾
<i>rveipcMinWidth</i>	203	MinWidth ⁽⁹¹⁴⁾
<i>rveipcAlignment</i>	204	Alignment ⁽⁹¹⁴⁾ *
<i>rveipcRemoveSpaceBelow</i>	205	RemoveSpaceBelow ⁽⁹¹⁵⁾ *
<i>rveipcUseTypoMetric</i>	206	UseTypeMetric

* the property is type-casted to Integer

Properties of Numbered sequences ⁽¹⁷⁷⁾

These properties are defined in RVSeqItem unit.

They identify properties of numbered sequences ⁽¹⁷⁷⁾ and items of all types inherited from them (footnotes ⁽¹⁸⁰⁾, endnotes ⁽¹⁷⁸⁾).

Constant	Value	Property
<i>rveipcSeqStartFrom</i>	300	StartFrom ⁽⁹²⁴⁾
<i>rveipcSeqReset</i>	301	Reset ⁽⁹²³⁾ *
<i>rveipcSeqNumberType</i>	302	NumberType ⁽⁹²³⁾ *, **

* the property is type-casted to Integer

** read-only for footnotes ⁽¹⁸⁰⁾, endnotes ⁽¹⁷⁸⁾, sidenotes ⁽¹⁸¹⁾ and text box items ⁽¹⁸³⁾.

Properties of Sidenotes ⁽¹⁸¹⁾

Position properties

These properties are defined in RVFloatingPos unit.

They identify sub-properties of BoxPosition ⁽⁹³⁵⁾ property of sidenotes ⁽¹⁸¹⁾ and text box items ⁽¹⁸³⁾.

Constant	Value	Property
<i>rveipcFloatHorizontalAnchor</i>	500	HorizontalAnchor ⁽⁹⁴¹⁾ *
<i>rveipcFloatHorizontalPositionKind</i>	501	HorizontalPositionKind ⁽⁹⁴²⁾ *
<i>rveipcFloatHorizontalOffset</i>	502	HorizontalOffset ⁽⁹⁴²⁾
<i>rveipcFloatHorizontalAlign</i>	503	HorizontalAlignment ⁽⁹⁴²⁾ *
<i>rveipcFloatVerticalAnchor</i>	504	VerticalAnchor ⁽⁹⁴⁵⁾ *
<i>rveipcFloatVerticalPositionKind</i>	505	VerticalPositionKind ⁽⁹⁴⁶⁾ *
<i>rveipcFloatVerticalOffset</i>	506	VerticalOffset ⁽⁹⁴⁶⁾
<i>rveipcFloatVerticalAlign</i>	507	VerticalAlignment ⁽⁹⁴⁶⁾ *
<i>rveipcFloatRelativeToCell</i>	508	RelativeToCell ⁽⁹⁴⁵⁾ *
<i>rveipcFloatPositionInText</i>	509	PositionInText ⁽⁹⁴⁴⁾ *

* the property is type-casted to Integer

Other properties

These properties are defined in RVFloatingBox unit.

They identify sub-properties of BoxProperties⁽⁹³⁵⁾ property of sidenotes⁽¹⁸¹⁾ and text box items⁽¹⁸³⁾.

Constant	Value	Property
<i>rveipcBoxWidth</i>	600	Width ⁽⁹³⁹⁾
<i>rveipcBoxHeight</i>	601	Height ⁽⁹³⁸⁾
<i>rveipcBoxWidthType</i>	602	WidthType ⁽⁹³⁹⁾ *
<i>rveipcBoxHeightType</i>	603	HeightType ⁽⁹³⁸⁾ *
<i>rveipcBoxBorderColor</i>	604	Border ⁽⁹³⁸⁾ .Color ⁽⁷⁴⁴⁾
<i>rveipcBoxBorderWidth</i>	605	Border ⁽⁹³⁸⁾ .Width ⁽⁷⁴⁴⁾
<i>rveipcBoxBorderInternalWidth</i>	606	Border ⁽⁹³⁸⁾ .InternalWidth ⁽⁷⁴⁴⁾
<i>rveipcBoxBorderStyle</i>	607	Border ⁽⁹³⁸⁾ .Style ⁽⁷⁴⁴⁾ *
<i>rveipcBoxBorderVisibleBorders_Left</i>	608	Border ⁽⁹³⁸⁾ .VisibleBorders ⁽⁷⁴⁴⁾ .Left*
<i>rveipcBoxBorderVisibleBorders_Top</i>	609	Border ⁽⁹³⁸⁾ .VisibleBorders ⁽⁷⁴⁴⁾ .Top*
<i>rveipcBoxBorderVisibleBorders_Right</i>	610	Border ⁽⁹³⁸⁾ .VisibleBorders ⁽⁷⁴⁴⁾ .Right*

Constant	Value	Property
<i>rveipcBoxBorderVisibleBorders_Bottom</i>	611	Border ⁽⁹³⁸⁾ .VisibleBorders ⁽⁷⁴⁴⁾ .Bottom*
<i>rveipcBoxBorderBorderOffsets_Left</i>	612	Border ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Left
<i>rveipcBoxBorderBorderOffsets_Top</i>	613	Border ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Top
<i>rveipcBoxBorderBorderOffsets_Right</i>	614	Border ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Right
<i>rveipcBoxBorderBorderOffsets_Bottom</i>	615	Border ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Bottom
<i>rveipcBoxColor</i>	616	Background ⁽⁹³⁸⁾ .Color ⁽⁷⁴²⁾
<i>rveipcBoxPadding_Left</i>	617	Background ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Left
<i>rveipcBoxPadding_Top</i>	618	Background ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Top
<i>rveipcBoxPadding_Right</i>	619	Background ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Right
<i>rveipcBoxPadding_Bottom</i>	620	Background ⁽⁹³⁸⁾ .BorderOffsets ⁽⁷⁴²⁾ .Bottom
<i>rveipcBoxVAlign</i>	621	VAlign ⁽⁹³⁹⁾ *
<i>rveipcBoxOpacity</i>	622	Background ⁽⁹³⁸⁾ .Opacity ⁽⁷⁴²⁾

* the property is type-casted to Integer

Properties of "Page number"⁽¹⁸⁵⁾ and "Page count"⁽¹⁸⁶⁾ fields

These properties are defined in RVFieldItems unit.

Constant	Value	Property
<i>rveipcPageNumberType</i>	600	NumberType ⁽⁹⁵²⁾ *

* the property is type-casted to Integer

Properties of mathematical expressions⁽¹⁸⁷⁾

These properties are defined in RVMathItem unit.

Constant	Value	Property
<i>rveipcFontSizeDouble</i>	200	FontSizeDouble ⁽⁹¹⁹⁾
<i>rveipcTextColor</i>	201	TextColor ⁽⁹²⁰⁾

Constant	Value	Property
<i>rveipcDisplayInline</i>	202	DisplayInline ⁽⁹¹⁸⁾ *

* the property is type-casted to Integer

See also

- *rvespc**** constants⁽¹⁰⁵⁶⁾

11.34 RVEMPTYTAG

Unit [VCL/FMX]: RVStyle / fmxRVStyle;

const

```
RVEMPTYTAG = '';
```

11.35 rvespc*** constants

These constants identify string properties of TRichView items⁽¹⁵⁸⁾.

They are used in the following methods:

- TRichView.GetExtraItemStrPropertyEx⁽³⁰⁰⁾
- TRichView.SetExtraItemStrPropertyEx⁽³⁵⁹⁾
- TRichViewEdit.SetItemExtraStrPropertyExEd⁽⁵⁶²⁾
- TRichViewEdit.GetCurrentItemExtraStrPropertyEx⁽⁵¹⁰⁾
- TRichViewEdit.SetCurrentItemExtraStrPropertyEx⁽⁵⁵⁷⁾

Properties of Label items⁽¹⁷⁶⁾

These properties are defined in RVLabelItem unit.

They identify properties of label items⁽¹⁷⁶⁾ and items of all types inherited from them (numbered sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, sidenotes⁽¹⁸¹⁾, references to notes⁽¹⁸⁴⁾, text boxes⁽¹⁸³⁾, page numbers⁽¹⁸⁵⁾, page counts⁽¹⁸⁶⁾).

Constant	Value	Property
<i>rvespcText</i>	200	Text ⁽⁹¹⁵⁾ *

* read-only for numbered sequences⁽¹⁷⁷⁾, footnotes⁽¹⁸⁰⁾, endnotes⁽¹⁷⁸⁾, sidenotes⁽¹⁸¹⁾, references to notes⁽¹⁸⁴⁾; ignored for text boxes⁽¹⁸³⁾

Properties of Numbered sequences⁽¹⁷⁷⁾

These properties are defined in RVSeqItem unit.

Constant	Value	Property
<i>rvespcSeqName</i>	300	SeqName ⁽⁹²⁴⁾
<i>rvespcSeqFormatString</i>	301	FormatString ⁽⁹²³⁾

Properties of mathematical expressions ⁽¹⁸⁷⁾

These properties are defined in RVMathItem unit.

Constant	Value	Property
<code>rvespcText</code>	200	Text ⁽⁹¹⁹⁾
<code>rvespcFontName</code>	201	FontName ⁽⁹¹⁸⁾

See also

- `rveipc***` constants ⁽¹⁰⁵¹⁾

11.36 RVGetSystemColor

Unit [VCL/FMX]: RVColorFuncs / fmxRVColorFuncs;

Returns value of

Syntax

type

```
TRVGetSystemColorFunction =  
    function (AControl: TControl; AColor: TRVColor (996)): 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 ⁽²⁵⁷⁾ = *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**.

11.37 RVGraphicHandler

Unit [VCL/FMX]: RVGrHandler / fmxRVGrHandler;

Syntax

```
var RVGraphicHandler: TRVGraphicHandler (972);
```

(introduced in version 15)

This variable returns an object used by TRichView components to manage graphic classes.

11.38 RVIsCustomURL

Unit [VCL/FMX]: RVFuncs / fmxRVFuncs;

Syntax

type

```
TCustomRVIsURLFunction = function (const Word: TRVUnicodeString1032): 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 RVIsURL⁹⁹¹ function.

See demos:

- Demos*\Assorted\Hypertext\URLs\

11.39 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.Options²⁴⁰, and *rvscParagraph* is included in RVVisibleSpecialCharacters¹⁰⁶¹ variable.

Value	Meaning
<i>rvpmStandard</i>	Pilcrow for paragraph breaks, bended arrow for line breaks
<i>rvpmArrows</i>	Bended arrow for paragraph breaks, arrow down for line breaks

Example:

Paragraph-1 ¶	Paragraph-1 ¶
Paragraph-2:·line-1 ¶	Paragraph-2:·line-1 ↓
Paragraph-2:·line-2 ¶	Paragraph-2:·line-2 ↓
rvpmStandard	rvpmArrows

See also:

- TRVStyle.SpecialCharactersColor⁽⁶⁵²⁾

11.40 rvs*** constants

There are three groups of **rvs***** constants:

- identifiers of types of RichView⁽²¹⁰⁾ items⁽⁸⁵⁾;
- indexes in the collection of text styles (RVStyle.TextStyles⁽⁶⁵⁴⁾) corresponding to the initial entries;
- values used by TRichView internally when saving and loading RVF⁽¹²⁴⁾ files (see RVF specification⁽¹⁴³⁾).

Identifiers of Item Types

Use the first group of constants to compare with values returned by RichView.GetItemStyle⁽³⁰²⁾ or TRichViewEdit.CurlItemStyle⁽⁴⁷³⁾.

Constant	Value	Item Type	Unit
<i>rvsBreak</i>	-1	break ⁽¹⁶⁷⁾ (horizontal line)	RVStyle
<i>rvsPicture</i>	-3	picture ⁽¹⁶³⁾	RVStyle
<i>rvsHotspot</i>	-4	hot spot ⁽¹⁷²⁾ (picture from ImageList - hyperlink)	RVStyle
<i>rvsComponent</i>	-5	inserted Delphi control ⁽¹⁶⁸⁾	RVStyle
<i>rvsBullet</i>	-6	bullet ⁽¹⁷⁰⁾ (picture from ImageList)	RVStyle
<i>rvsHotPicture</i>	-10	hot picture ⁽¹⁶⁶⁾ (picture - hyperlink)	RVStyle
<i>rvsListMarker</i>	-11	list marker ⁽¹⁷⁴⁾ (bullet or numbering of paragraphs)	RVStyle
<i>rvsTab</i>	-12	tabulator ⁽¹⁶²⁾	RVStyle
<i>rvsTable</i>	-60	table ⁽¹⁹⁰⁾	RVTable
<i>rvsLabel</i>	-200	label ⁽¹⁷⁶⁾ (non-text item looking like text)	RVLabelItem
<i>rvsSequence</i>	-202	numbered sequence ⁽¹⁷⁷⁾	RVSeqItem
<i>rvsFootnote</i>	-203	footnote ⁽¹⁸⁰⁾	RVNote
<i>rvsEndnote</i>	-204	endnote ⁽¹⁷⁸⁾	RVNote
<i>rvsSidenote</i>	-206	sidenotes ⁽¹⁸¹⁾	RVSidenote
<i>rvsTextBox</i>	-207	text boxes ⁽¹⁸³⁾	RVSidenote
<i>rvsNoteReference</i>	-205	reference to the parent note ⁽¹⁸⁴⁾	RVNote
<i>rvsPageNumber</i>	-208	"page number" field ⁽¹⁸⁵⁾	RVFieldItems

Constant	Value	Item Type	Unit
<i>rvsPageCount</i>	-209	"page count" field ⁽¹⁸⁶⁾	RVFieldItems
<i>rvsMathEquation</i>	-210	mathematical expression ⁽¹⁸⁷⁾	RVMathItem

Item types included in **Report Workshop**:

Constant	Value	Item Type	Unit
<i>rvsReportTable</i>	-61	report table	RVReportTable
<i>rvsShape</i>	-211	geometric shape	RVReportShapelItem

Types for demo items (in TRichView\Demos\Addins\)

Constant	Value	Item Type	Unit
<i>rvsBlendBitmap</i>	-50	semitransparent bitmap image	RVBlendBitmap
<i>rvsHotBlendBitmap</i>	-51	hypertext semitransparent bitmap image	RVBlendBitmap
<i>rvsChart</i>	-1001	TChart item	ChartItem
<i>rvsCombo</i>	-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 ⁽⁸⁵⁾.

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
<i>rvsNormal</i>	0	Normal text
<i>rvsHeading</i>	1	Heading 1
<i>rvsSubheading</i>	2	Heading 2
<i>rvsKeyword</i>	3	Keywords
<i>rvsJump1</i>	4	Hypertext 1
<i>rvsJump2</i>	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
<i>rvsDefStyle</i>	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 ⁽⁷²⁹⁾ will be used. This feature is deprecated, use *StyleTemplates* ⁽⁹⁸⁾ instead.

This value also has a special meaning in *TRichView.OnReadField* ⁽³⁸⁶⁾ event.

11.41 RVThumbnailMaker

Unit [VCL/FMX]: *RVThumbMaker* / *fmxRVThumbMaker*;

Syntax

```
var RVThumbnailMaker: TRVThumbnailMaker (979);
```

(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 ⁽¹²⁰⁾

11.42 RVVisibleSpecialCharacters

Unit [VCL/FMX]: *RVStyle* / *fmxRVStyle*;

Syntax

type

```
TRVSpecialCharacter = (rvscSpace, rvscNBSP, rvscTab, rvscParagraph,  
rvscSoftHyphen, rvscPlaceholders, rvscFloatingLines);
```

```
TRVSpecialCharacters = set of TRVSpecialCharacter;
```

const

```
RVVisibleSpecialCharacters: TRVSpecialCharacters =  
[rvscSpace..rvscFloatingLines];
```

(introduced in version 10)

Lists characters which will be shown when *rvoShowSpecialCharacters* is included in *TCustomRichView.Options* ⁽²⁴⁰⁾.

Value	Meaning
<i>rvscSpace</i>	If included, space characters are shown (as middle dots). Hidden spaces at the ends of lines are not shown, see

Value	Meaning
	RichViewShowGhostSpaces ¹⁰⁴⁸
<i>rvscNBSP</i>	If included, non breaking space characters are shown (as circles)
<i>rvscTab</i>	If included, tab characters are shown (as arrows)
<i>rvscParagraph</i>	If included, end-of-paragraph and end-of-line marks are shown (as pilcrows and arrows, respectively)
<i>rvscSoftHyphen</i>	If included, soft hyphen characters are shown in Unicode text.
<i>rvscParagraphAttrs</i>	If included together with <i>rvscParagraph</i> , additional icons are drawn next to end-of-paragraph marks: <ul style="list-style-type: none"> ▪ triangle-arrow showing paragraph direction⁷²¹ (if defined); ▪ rectangle if any non-default printing option⁷²³ is defined (never displayed for headings 1...9) ▪ heading level (1...9) ▪ text flow clearing²²⁵ (for the next paragraph or this line break)
<i>rvscPlaceholders</i>	If included, special marks are painted at places of insertion of text box items ¹⁸³ . See also RVAlwaysShowPlaceholders ¹⁰⁵¹ .
<i>rvcFloatingLines</i>	If included, floating lines are displayed for left-and right-aligned items ¹⁰³³ , see TRVStyle.FloatingLineColor ⁶³⁸

See also:

- RVParagraphMarks¹⁰⁵⁸
- TRVStyle.SpecialCharactersColor⁶⁵²

12 How to...

How to...

(printing)

- Change page size and orientation for printing RichView¹⁰⁶³
- Draw page numbers, headers and footers¹⁰⁶³

(editing)

- Implement commands like "make bold", "apply font", etc.¹⁰⁶³
- Implement Find and Replace dialogs¹⁰⁶⁴
- Switch insert/overtyping mode¹⁰⁶⁴
- Move the caret to the beginning or to the end of document in editor¹⁰⁶⁵
- Make Unicode editor¹⁰⁶⁶
- Make a plain text editor¹⁰⁶⁶
- Implement smart indents¹⁰⁶⁸

(miscellaneous)

- Use third-party graphic classes with RichView¹⁰⁶⁸
- Combine several documents in one¹⁰⁶⁸
- Remove all formatting¹⁰⁷¹

12.1 change page size and orientation

TRVPrint⁽⁷⁵⁹⁾ 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⁽⁷⁷⁴⁾ 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⁽⁷⁵⁹⁾ component by the methods SetHeader⁽⁷⁷⁹⁾ and SetFooter⁽⁷⁷⁸⁾.

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⁽¹⁸⁵⁾ and page counts⁽¹⁸⁶⁾.

Drawing headers and footers yourself

If you need different headers for different pages, you need to draw them yourself, using OnPagePrepaint⁽⁷⁸²⁾ 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⁽⁸⁰⁴⁾ 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⁽⁴⁹²⁾.

Example:

- Demos*\Editors\Editor 2\

Alternatively, you can use [RichViewActions](#).

12.4 implement Find and Replace dialogs

Find

The key method is `TRichView.SearchText`⁽³⁴⁶⁾ or `TRichViewEdit.SearchText`⁽⁵⁴³⁾.

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`⁽⁹⁸⁷⁾ or `GetRVESearchOptions`⁽⁹⁸⁶⁾.

Replace

Use `TRichViewEdit.SearchText`⁽⁵⁴³⁾ to select the old string.

Then use `InsertText`⁽⁵³²⁾ 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/overtyping 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(349);
end;

procedure TForm1.RichViewEdit1KeyPress(Sender: TObject; var Key: Char);
var rve: TCustomRichViewEdit(461);
    ItemNo, Offs: Integer;
begin
  if IgnoreNextChar then
  begin
    IgnoreNextChar := False;
```

```

    exit;
end;
IgnoreNextChar := False;
if not ((Key=#9) or (Key>=' ')) then
    exit;
rve := RichViewEdit1.TopLevelEditor481;
if rve.SelectionExists349 then
    exit;
ItemNo := rve.CurItemNo472;
Offs := rve.OffsetInCurItem479;
if (Offs>=rve.GetOffsAfterItem308(ItemNo)) then
begin
    if (ItemNo+1<rve.ItemCount237) and
        not rve.IsFromNewLine317(ItemNo+1) then
        begin
            inc(ItemNo);
            Offs := rve.GetOffsBeforeItem309(ItemNo);
        end
    else
        exit;
    end;
rve.SetSelectionBounds365(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.MoveCaret534(rvcmTop);
```

Moving to the end:

```
MyRichViewEdit.MoveCaret534(rvcmBottom);
```

Option 2

The caret is always at the end of selection.

Selection can be changed using method TRichViewEdit.SetSelectionBounds³⁶⁵.

Moving to the beginning:

```

var
    ItemNo, Offs: Integer;
...
ItemNo := 0;
Offs := MyRichViewEdit.GetOffsBeforeItem309(ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo, Offs, ItemNo, Offs);

```

Moving to the end:

```

var
  ItemNo, Offs: Integer;
...
ItemNo := MyRichViewEdit.ItemCount(237) - 1;
Offs := MyRichViewEdit.GetOffsAfterItem(308)(ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo, Offs, ItemNo, Offs);

```

See also:

- TRichViewEdit.CurItemNo⁽⁴⁷²⁾ and OffsetInCurItem⁽⁴⁷⁹⁾;
- Example how to move caret to the beginning of paragraph⁽⁵⁹²⁾.

12.7 make Unicode editor

See in the topic Unicode in TRichView⁽¹³⁰⁾.

(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⁽⁴⁶¹⁾

- Exclude all but text from AcceptDragDropFormat⁽⁴⁷⁰⁾ property.
- Process OnPaste⁽⁵⁸⁴⁾ 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(532)(s, False);
  end;
  DoDefault := False;
end;

```

How to make ONE LINE plain text editor

- Exclude all but text from AcceptDragDropFormat⁽⁴⁷⁰⁾ property.
- Process OnPaste⁽⁵⁸⁴⁾ event (see below).
- Process OnOleDrop⁽⁵⁸¹⁾ event (see below).
- Disable all UI commands changing text and paragraph attributes or inserting multiline text.
- Set WordWrap⁽²⁵⁸⁾ property = *False*.
- Include *rvDoNotWantReturns* in EditorOptions⁽⁴⁷⁵⁾.
- Set VSmallStep⁽²⁵⁷⁾ property = 1, before calling Format⁽²⁸⁸⁾.
- If you do not want tabulators as well, set RVStyle.SpacesInTab⁽⁶⁵²⁾ = 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(210);
  const DataObject: IDataObject; Shift: TShiftState; X, Y: Integer;
  PossibleDropEffects: TRVOleDropEffects(1015);
  var DropEffect: TRVOleDropEffect(1015); 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(532)(MakeOneLineString(s), False);
    end;

```

```
ReleaseStgMedium(StgMedium);  
end;  
  
procedure TMyForm.MyRichViewEditPaste(Sender: TCustomRichViewEdit461;  
  var DoDefault: Boolean);  
var s: String;  
begin  
  if Clipboard.HasFormat(CF_TEXT) then  
  begin  
    s := Clipboard.AsText;  
    Sender.InsertText532(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 FirstIndent⁷²² and LeftIndent⁷²² of paragraph, if **Tab**/**Backspace** is pressed when the caret is at the beginning of paragraph. First, it changes FirstIndent⁷²². Next, if FirstIndent⁷²² is already changed, it changes LeftIndent⁷²². 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.InsertRVFFromStream³¹⁶.

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 TRVGraphicFM⁹⁷⁰.

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`⁽²⁷²⁾ or `InsertPicture`⁽⁵²⁶⁾.

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⁽¹¹⁹⁾, assign `TCustomRichView.AnimationMode`⁽²²²⁾ = `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⁽¹¹⁹⁾

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 `TRVGraphicHandler.SetDefaultGraphicClass`⁽⁹⁷²⁾.

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`⁽¹⁰⁵⁷⁾.`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`⁽¹⁰⁵⁷⁾.`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⁽¹⁵⁴⁾).

Specifying PNG class

Specify graphic class for PNG images using `RVGraphicHandler`⁽¹⁰⁵⁷⁾.`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 `RVGraphicHandler`⁽¹⁰⁵⁷⁾.`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

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⁽¹⁰⁶⁸⁾), including animations⁽¹¹⁹⁾. For additional ideas about emoticons, see the demo in Demos*\Assorted\Graphics\Emoticons\.

URL detection use RVIsURL and RVIsEmail⁽⁹⁹¹⁾ functions. You can use RVIsCustomURL⁽¹⁰⁵⁸⁾ 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⁽⁶⁵⁴⁾ and paragraph styles⁽⁶⁴⁸⁾ to 0s. Optionally, it can keep hyperlinks (they are converted to the hypertext style with the lowest index).

Optionally, it can remove list markers⁽¹⁷⁴⁾ as well.

This code changes text items⁽¹⁶¹⁾, tabulators⁽¹⁶²⁾ and list markers. All other item types remains unchanged.

Note 1: if the document contains both Unicode⁽¹³⁰⁾ 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(957);
  RemoveMarkers, KeepLinks: Boolean);
var
  i, HypertextStyleNo: Integer;
  {.....}
procedure DoRemoveFormatting(RVData: TCustomRVData(957));
var
  i, r, c, StyleNo, StyleNoTo: Integer;
  PB: Boolean;
  table: TRVTableItemInfo(846);
begin
  for i := RVData.ItemCount(237)-1 downto 0 do
  begin
    RVData.GetItem(296)(i).ParaNo := 0;
    StyleNo := RVData.GetItemStyle(302)(i);
    case StyleNo of
      rvsTable:
        begin
          table := TRVTableItemInfo(846)(RVData.GetItem(296)(i));
          for r := 0 to table.RowCount(864)-1 do
            for c := 0 to table.ColCount(858)-1 do
              if table.Cells(857)[r,c]<>nil then
                DoRemoveFormatting(table.Cells(857)[r,c].GetRVData(960));
            end;
          rvsListMarker:
            if RemoveMarkers then begin
              PB := RVData.PageBreaksBeforeItems(244)[i];
              RVData.DeleteItems(283)(i, 1);
              if i<RVData.ItemCount(237) then
                begin
                  RVData.GetItem(296)(i).SameAsPrev := False;
                  RVData.GetItem(296)(i).PageBreakBefore := PB;
                end;
              end;
            end;
          rvsTab:
            if RVData.GetRVStyle.TextStyles(654)[
              TRVTabItemInfo(RVData.GetItem(296)(i)).TextStyleNo].Jump(714) and
              KeepLinks then
                TRVTabItemInfo(RVData.GetItem(296)(i)).TextStyleNo :=

```

```

        HypertextStyleNo
    else
    begin
        TRVTabItemInfo(RVData.GetItem296(i)).TextStyleNo := 0;
        RVData.SetItemTag360(i, '');
    end;
1..MaxInt:
    begin
        if KeepLinks and
            RVData.GetRVStyle.TextStyles654[StyleNo].Jump714 then
            StyleNoTo := HypertextStyleNo
        else
        begin
            StyleNoTo := 0;
            RVData.SetItemTag360(i, '');
        end;
        RVData.GetItem296(i).StyleNo := StyleNoTo;
    end;
end;
end;
end;
{.....}
begin
    HypertextStyleNo := 0;
    if KeepLinks then
        for i := 0 to RVData.GetRVStyle.TextStyles654.Count-1 do
            if RVData.GetRVStyle.TextStyles654[i].Jump714 then
                begin
                    HypertextStyleNo := i;
                    break;
                end;
        end;
    DoRemoveFormatting(RVData);
end;

```

How to use:

```

RemoveFormatting(RichViewEdit1.RVData247, True, True);
// from RVNormalize.pas, from RichViewActions
NormalizeRichView(RichViewEdit1.RVData247);
RichViewEdit1.DeleteUnusedStyles286(True, True, True);
RichViewEdit1.ClearUndo501;
RichViewEdit1.Format288;

```

More information: <https://www.trichview.com/forums/viewtopic.php?t=2392>

Index

- A -

- AcceptDragDropFormats 470
 - TRichViewEdit 470
- AcceptPasteFormats 470
 - TRichViewEdit 470
- Active 784
 - TRVirtualPrinterProperties 784
- ActualCurTextStyleNo 472
 - TCustomRichViewEdit 472
- Add 270, 678, 684, 687, 749
 - TFontInfos 678
 - TParaInfos 684
 - TRichView 270
 - TRVListInfos 687
 - TRVListLevelCollection 749
- AddBreak 262
 - TRichView 262
- AddBreakEx 262
 - TRichView 262
- AddBreakExTag 262
 - TRichView 262
- AddBreakTag 262
 - TRichView 262
- AddBullet 263
 - TRichView 263
- AddBulletEx 263
 - TRichView 263
- AddBulletExTag 263
 - TRichView 263
- AddCheckpoint 264
 - TRichView 264
- AddCheckpointTag 264
 - TRichView 264
- AddControl 265
 - TRichView 265
- AddControlEx 265
 - TRichView 265
- AddControlExTag 265
 - TRichView 265
- AddFmt 266
 - TRichView 266
- AddFont 678
 - TFontInfos 678
- AddFontEx 678
 - TFontInfos 678
- AddFrom 746
 - TRVTabInfos 746
- AddHotPicture 267
 - TRichView 267
- AddHotPictureTag 267
 - TRichView 267
- AddHotspot 268
 - TRichView 268
- AddHotspotEx 268
 - TRichView 268
- AddHotspotExTag 268
 - TRichView 268
- AddItem 269
 - TRichView 269
- AddNamedCheckpoint 264
 - TRichView 264
- AddNamedCheckpointEx 264
 - TRichView 264
- AddNamedCheckpointExTag 264
 - TRichView 264
- AddNL 270
 - TRichView 270
- AddNLATag 270
 - TRichView 270
- AddNLTag 270
 - TRichView 270
- AddNLWTag 270
 - TRichView 270
- AddPicture 272
 - TRichView 272
- AddPictureEx 272
 - TRichView 272
- AddPictureExTag 272
 - TRichView 272
- AddSpellingMenuItems 488
 - TRichViewEdit 488
- AddTab 273
 - TRichView 273
- AddTag 270
 - TRichView 270
- AddTextNL 274
 - TRichView 274
- AddTextNLA 274
 - TRichView 274
- AddTextNLW 274
 - TRichView 274
- AddTextStyle 657
 - TRVStyle 657
- AddToAutoreplaceList 788

- AddToAutoreplaceList 788
 - TRVSpellChecker 788
 - AddToDictionary 789
 - TRVSpellChecker 789
 - AddToIgnoreList 789
 - TRVSpellChecker 789
 - AddXXX methods of TRichView 102
 - AdjustControlPlacement 489
 - TRichViewEdit 489
 - AdjustControlPlacement2 489
 - TRichViewEdit 489
 - Align 747
 - TRVTabInfo 747
 - Alignment 719, 914
 - TCustomRVParaInfo 719
 - TRVLabelItemInfo 914
 - AllNumbered 733
 - TRVListInfo 733
 - AllowMarkdown 422, 598, 611
 - TDBRichView 598
 - TDBRichViewEdit 611
 - TRVDocumentProperty 422
 - AllowSelection 222
 - TRichView 222
 - AllowTransparentDrawing 806
 - TRVReportHelper 806
 - Animation in TRichView 119
 - AnimationMode 222
 - TRichView 222
 - AppendFrom 275
 - TRichView 275
 - AppendRVFFromStream 276
 - TRichView 276
 - ApplyListStyle 490
 - TRichViewEdit 490
 - ApplyParaStyle 490
 - TRichViewEdit 490
 - ApplyParaStyleConversion 491
 - TRichViewEdit 491
 - ApplyParaStyleTemplate 491
 - TRichViewEdit 491
 - ApplyStyleConversion 492
 - TRichViewEdit 492
 - ApplyStyleTemplate 493
 - TRichViewEdit 493
 - ApplyTextStyle 494
 - TRichViewEdit 494
 - ApplyTextStyleTemplate 494
 - TRichViewEdit 494
 - ApplyToParaStyle 738
 - TRVStyleTemplate 738
 - ApplyToTextStyle 739
 - TRVStyleTemplate 739
 - Assign 711, 715, 726, 730
 - TCustomRVFontInfo 711
 - TCustomRVParaInfo 726
 - TFontInfo 715
 - TParaInfo 730
 - AssignDocParameters 771
 - TRVPrint 771
 - AssignProperties 870
 - TRVTableItemInfo 870
 - AssignSoftPageBreaks 277
 - TCustomRichView 277
 - AssignSource 771
 - TRVPrint 771
 - AssignStyleTemplates 690
 - TRVStyleTemplateCollection 690
 - AssignTo 678, 684, 687, 711
 - TCustomRVFontInfo 711
 - TFontInfos 678
 - TParaInfos 684
 - TRVListInfos 687
 - AssignToRVPrint 977
 - TRVReportHelperWithHeaderFooters 977
 - AssignToStrings 690
 - TRVStyleTemplateCollection 690
 - Author 416
 - TRVDocParameters 416
 - AutoDeleteUnusedStyles 423, 599, 611
 - TDBRichView 599
 - TDBRichViewEdit 611
 - TRVDocumentProperty 423
 - AutoDisplay 599, 612
 - TDBRichView 599
 - TDBRichViewEdit 612
 - AutoHideTableGridLines 451
 - TRVRTFReaderProperties 451
 - AvailableLanguages 786
 - TRVSpellChecker 786
- B -**
- BackColor 698
 - TCustomRVFontInfo 698
 - Background 721, 938
 - TCustomRVParaInfo 721
 - TRVBoxProperties 938
 - BackgroundBitmap 222
 - TRichView 222

- BackgroundImage 849, 896
 - TRVTableCellData 896
 - TRVTableItemInfo 849
- BackgroundImageFileName 850, 897
 - TRVTableCellData 897
 - TRVTableItemInfo 850
- BackgroundStyle 223, 850, 897
 - TRichView 223
 - TRVTableCellData 897
 - TRVTableItemInfo 850
- BasePathLinks 425, 446, 451
 - TRVHTMLReaderProperties 425
 - TRVRTFReaderProperties 451
- BeginItemModify 495
 - TRichViewEdit 495
- BeginOleDrag 278
 - TRichView 278
- BeginUndoCustomGroup 496
 - TRichViewEdit 496
- BeginUndoCustomGroup2 496
 - TRichViewEdit 496
- BeginUndoGroup 497
 - TRichViewEdit 497
- BeginUndoGroup2 497
 - TRichViewEdit 497
- BeginUpdate 498
 - TRichViewEdit 498
- BestDPI 765
 - TRVPrint 765
- BestHeight 898
 - TRVTableCellData 898
- BestWidth 850, 898
 - TRVTableCellData 898
 - TRVTableItemInfo 850
- BiDiMode 132, 224, 699, 721
 - TCustomRVFontInfo 699
 - TCustomRVParaInfo 721
 - TRichView 224
- Bidirectional text in TRichView 132
- BlockQuoteBackColor 438
 - TRVMarkdownDefaultItemProperties 438
- Border 721, 938
 - TCustomRVParaInfo 721
 - TRVBoxProperties 938
- BorderColor 851, 899
 - TRVTableCellData 899
 - TRVTableItemInfo 851
- BorderHSpacing 851
 - TRVTableItemInfo 851
- BorderLightColor 852, 899
 - TRVTableCellData 899
 - TRVTableItemInfo 852
- BorderOffsets 742, 743
 - TRVBackgroundRect 742
 - TRVBorder 743
- Borders and background of paragraphs in TRichView 97
- BorderStyle 816, 852
 - TRVScroller 816
 - TRVTableItemInfo 852
- BorderVSpacing 853
 - TRVTableItemInfo 853
- BorderWidth 853
 - TRVTableItemInfo 853
- BottomMargin 225, 416
 - TRichView 225
 - TRVDocParameters 416
- BoxPosition 935
 - TRVSidenoteItemInfo 935
- BoxProperties 935
 - TRVSidenoteItemInfo 935
- BreakColor 438
 - TRVMarkdownDefaultItemProperties 438
- Breaks 167
- Building TRichView document 102
- Bullets 170
- ButtonType 589
 - TRVSmartPopupProperties 589

- C -

- CachePageImage 628
 - TRVPrintPreview 628
- CanChange 498
 - TRichViewEdit 498
- CanMergeCells 870
 - TRVTableItemInfo 870
- CanMergeSelectedCells 871
 - TRVTableItemInfo 871
- CanPaste 498
 - TRichViewEdit 498
- CanPasteHTML 499
 - TRichViewEdit 499
- CanPasteRTF 499
 - TRichViewEdit 499
- CanPasteRVF 499
 - TRichViewEdit 499
- CanReuseNumberedLists 452
 - TRVRTFReaderProperties 452
- CanSavePDF 827

- CanSavePDF 827
 - TCustomRVPrint 827
- Cell Merging 198
- CellBorderColor 853
 - TRVTableItemInfo 853
- CellBorderLightColor 854
 - TRVTableItemInfo 854
- CellBorderStyle 854
 - TRVTableItemInfo 854
- CellBorderWidth 855
 - TRVTableItemInfo 855
- CellHPadding 855
 - TRVTableItemInfo 855
- CellHSpacing 856
 - TRVTableItemInfo 856
- CellOverrideColor 856
 - TRVTableItemInfo 856
- CellPadding 856
 - TRVTableItemInfo 856
- Cells 857
 - TRVTableItemInfo 857
- CellVPadding 857
 - TRVTableItemInfo 857
- CellVSpacing 857
 - TRVTableItemInfo 857
- CFRV_HTML 1037
- CFRV_RTF 1037
- CFRV_RVF 1037
- CFRV_URL 1037
- Change 499, 621
 - TDBRichViewEdit 621
 - TRichViewEdit 499
- Changed 871
 - TRVTableItemInfo 871
- ChangeListLevels 500
 - TRichViewEdit 500
- ChangeStyleTemplates 500
 - TRichViewEdit 500
- CharScale 699
 - TCustomRVFontInfo 699
- Charset 700
 - TCustomRVFontInfo 700
- CharsetForUnicode 452
 - TRVRTFReaderProperties 452
- CharSpacing 700
 - TCustomRVFontInfo 700
- CheckpointColor 634
 - TRVStyle 634
- CheckpointEvColor 634
 - TRVStyle 634
- Checkpoints in TRichView 87
- CheckpointsPrefix 430
 - TRVHTMLSaveProperties 430
- CheckSpelling 472
 - TRichViewEdit 472
- Clear 279, 501, 827
 - TCustomRVPrint 827
 - TRichView 279
 - TRichViewEdit 501
- ClearLeft 225
 - TRichView 225
- ClearLiveSpellingResults 279
 - TRichView 279
- ClearParaFormat 690
 - TRVStyleTemplateCollection 690
- ClearRight 226
 - TRichView 226
- ClearSoftPageBreaks 279
 - TRichView 279
- ClearTextFlow 501
 - TRichViewEdit 501
- ClearTextFormat 690
 - TRVStyleTemplateCollection 690
- ClearUndo 501
 - TRichViewEdit 501
- ClickMode 836
 - TCustomRVPrintPreview 836
- ClientToDocument 280
 - TRichView 280
- ClipMargins 765
 - TRVPrint 765
- CodeBlockBackColor 439
 - TRVMarkdownDefaultItemProperties 439
- CodePage 423, 599, 612
 - TDBRichView 599
 - TDBRichViewEdit 612
 - TRVDocumentProperty 423
- ColBandSize 858
 - TRVTableItemInfo 858
- ColCount 858
 - TRVTableItemInfo 858
- Color 226, 589, 635, 701, 742, 744, 836, 858, 900
 - TCustomRVFontInfo 701
 - TCustomRVPrintPreview 836
 - TRichView 226
 - TRVBackgroundRect 742
 - TRVBorder 744
 - TRVSmartPopupProperties 589
 - TRVStyle 635
 - TRVTableCellData 900

- Color 226, 589, 635, 701, 742, 744, 836, 858, 900
 - TRVTableItemInfo 858
- ColorMode 822
 - TCustomRVPrint 822
- Colors and layout of tables 193
- ColSpan 900
 - TRVTableCellData 900
- Comments 417
 - TRVDocParameters 417
- Component Editor for TRichView 218
- ContinuousPrint 772
 - TRVPrint 772
- Controls 168
- Controls - Documents - Items 136
- ConvertDocToDifferentUnits 280
 - TRichView 280
- ConvertDocToEMU 280
 - TRichView 280
- ConvertDocToPixels 280
 - TRichView 280
- ConvertDocToTwips 280
 - TRichView 280
- ConvertHighlight 452
 - TRVRTFReaderProperties 452
- ConvertSymbolFonts 453
 - TRVRTFReaderProperties 453
- ConvertToDifferentUnits 657
 - TRVStyle 657
- ConvertToEMU 657
 - TRVStyle 657
- ConvertToHotPicture 501
 - TRichViewEdit 501
- ConvertToPicture 502
 - TRichViewEdit 502
- ConvertToPixels 657
 - TRVStyle 657
- ConvertToTwips 657
 - TRVStyle 657
- ConvertToUnit 421
 - TRVDocParameters 421
- ConvertToUnits 773
 - TRVPrint 773
- Copy 281
 - TRichView 281
- CopyDef 281
 - TRichView 281
- CopyImage 281
 - TRichView 281
- CopyRTF 282
 - TRichView 282
- CopyRVF 282
 - TRichView 282
- CopyText 283
 - TRichView 283
- CopyTextA 283
 - TRichView 283
- CopyTextW 283
 - TRichView 283
- Count 803
 - TRVOfficeCnvList 803
- CPEventKind 227
 - TRichView 227
- Create 283, 502, 604, 621, 657, 678, 684, 711, 716, 727, 731, 773, 799, 820, 841, 872, 916, 920, 924, 930, 932, 936, 948, 951, 953, 955, 977
 - TCustomRVFontInfo 711
 - TCustomRVParaInfo 727
 - TCustomRVPrintPreview 841
 - TDBRichView 604
 - TDBRichViewEdit 621
 - TFontInfo 716
 - TFontInfos 678
 - TParaInfo 731
 - TParaInfos 684
 - TRichView 283
 - TRichViewEdit 502
 - TRVEndnoteItemInfo 930
 - TRVFootnoteItemInfo 932
 - TRVLabelItemInfo 916
 - TRVMathItemInfo 920
 - TRVNoteReferenceItemInfo 951
 - TRVOfficeConverter 799
 - TRVPageCountItemInfo 955
 - TRVPageNumberItemInfo 953
 - TRVPrint 773
 - TRVReportHelperWithHeaderFooters 977
 - TRVScroller 820
 - TRVSeqItemInfo 924
 - TRVSidenoteItemInfo 936
 - TRVStyle 657
 - TRVTableItemInfo 872
 - TRVTextBoxItemInfo 948
- CreateEx 872, 916, 925, 927, 930, 933, 936, 948, 951, 954, 955
 - TCustomRVNoteItemInfo 927
 - TRVEndnoteItemInfo 930
 - TRVFootnoteItemInfo 933
 - TRVLabelItemInfo 916
 - TRVNoteReferenceItemInfo 951
 - TRVPageCountItemInfo 955
 - TRVPageNumberItemInfo 954
 - TRVSeqItemInfo 925

- CreateEx 872, 916, 925, 927, 930, 933, 936, 948, 951, 954, 955
 - TRVSidenoteItemInfo 936
 - TRVTableItemInfo 872
 - TRVTextBoxItemInfo 948
 - crJump 142
 - crRVZoomIn 142
 - crRVZoomOut 142
 - CSSOptions 430
 - TRVHTMLSaveProperties 430
 - CSSSavingType 431
 - TRVHTMLSaveProperties 431
 - CurlItemNo 472
 - TRichViewEdit 472
 - CurlItemStyle 473
 - TRichViewEdit 473
 - CurParaStyleNo 473
 - TRichViewEdit 473
 - CurrentItemColor 635
 - TRVStyle 635
 - CurrentLanguage 789
 - TRVSpellChecker 789
 - Cursor 227, 914
 - TRichView 227
 - TRVLabelItemInfo 914
 - Cursors in TRichView 142
 - CurTextStyleNo 474
 - TRichViewEdit 474
 - Custom drawing in TRichView 141
 - Custom item types in TRichView 189
 - CustomCaretInterval 474
 - TRichViewEdit 474
 - CutDef 502
 - TRichViewEdit 502
- D -**
- DarkMode 228, 823
 - TCustomRVPrint 823
 - TRichView 228
 - DarkModeUI 837
 - TCustomRVPrintPreview 837
 - DataField 600, 613
 - TDBRichView 600
 - TDBRichViewEdit 613
 - DataSource 600, 613, 812
 - TDBRichView 600
 - TDBRichViewEdit 613
 - TRVDataSourceLink 812
 - DefaultOStyle 431
 - TRVHTMLSaveProperties 431
 - DefaultBulletFontName 963
 - TRVDefaultLoadProperties 963
 - DefaultFontName 964
 - TRVDefaultLoadProperties 964
 - DefaultFontSizeDouble 964
 - TRVDefaultLoadProperties 964
 - DefaultFontSizesDouble 964
 - TRVDefaultLoadProperties 964
 - DefaultItemProperties 444
 - TRVMarkdownProperties 444
 - DefaultMonoSpacedFontName 965
 - TRVDefaultLoadProperties 965
 - DefaultPictureVAlign 475
 - TRichViewEdit 475
 - DefaultProperties 965
 - TRVDefaultLoadProperties 965
 - DefaultRVFPixelsPerInch 1042
 - DefaultUnicodeSymbolFontName 453, 965
 - TRVDefaultLoadProperties 965
 - DefCodePage 635
 - TRVStyle 635
 - DefStyleName 688
 - TRVStyleTemplateCollection 688
 - DefStyleNo 729
 - TParaInfo 729
 - DefTabWidth 636
 - TRVStyle 636
 - DefUnicodeStyle 636
 - TRVStyle 636
 - DeleteCols 873
 - TRVTableItemInfo 873
 - DeleteEmptyCols 873
 - TRVTableItemInfo 873
 - DeleteEmptyRows 873
 - TRVTableItemInfo 873
 - DeleteItems 283
 - TRichView 283
 - DeleteList 746
 - TRVTabInfos 746
 - DeleteMarkedStyles 284
 - TRichView 284
 - DeleteParas 285
 - TRichView 285
 - DeleteRows 874
 - TRVTableItemInfo 874
 - DeleteSection 285
 - TRichView 285
 - DeleteSelectedCols 874
 - TRVTableItemInfo 874

DeleteSelectedRows 875
 TRVTableItemInfo 875

DeleteSelection 503
 TRichViewEdit 503

DeleteTextStyle 658
 TRVStyle 658

DeleteUnusedStyles 286
 TRichView 286

Delimiters 228
 TRichView 228

Deselect 286, 875
 TRichView 286
 TRVTableItemInfo 875

Destroy 286, 503, 604, 621, 658, 799, 876
 TDBRichView 604
 TDBRichViewEdit 621
 TRichView 286
 TRichViewEdit 503
 TRVOfficeConverter 799
 TRVStyle 658
 TRVTableItemInfo 876

DialogShown 787
 TRVSpellChecker 787

DifferentUnitsToUnits 659
 TRVStyle 659

DisabledFontColor 637
 TRVStyle 637

DisplayInline 918
 TRVMathItemInfo 918

DocObjects 229
 TRichView 229

DocParameters 230
 TRichView 230

DocProperties 231
 TRichView 231

Document 232, 926
 TCustomRVNoteltemInfo 926
 TRichView 232

DocumentHeight 233
 TRichView 233

DocumentPixelsPerInch 233
 TRichView 233

DocumentToClient 280
 TRichView 280

DocXErrorCode 454
 TRVRTFReaderProperties 454

DoInPaletteMode 234
 TRichView 234

Drag and drop in TRichView 118

Drag&Drop in TRichView 118

Draw 716
 TFontInfo 716

DrawButton 981

DrawControl 981

DrawEdit 981

DrawMemo 981

DrawPage 773, 808
 TRVPrint 773
 TRVReportHelper 808

DrawPageAt 773, 808
 TRVPrint 773
 TRVReportHelper 808

DrawPanel 981

DrawPreview 773
 TRVPrint 773

- E -

Edit 959
 TCustomRVData 959

EditCell 876
 TRVTableItemInfo 876

Editing Cells 197

Editor 812
 TRVDataSourceLink 812

EditorOptions 475
 TRichViewEdit 475

EmptyWidth 701
 TCustomRVFontInfo 701

EMUToUnits 659
 TRVStyle 659

Encoding 431
 TRVHTMLSaveProperties 431

EncodingStr 432
 TRVHTMLSaveProperties 432

EndAt 823
 TCustomRVPrint 823

EndItemModify 503
 TRichViewEdit 503

EndnoteNumbering 637
 TRVStyle 637

Endnotes 178

EndUndoCustomGroup2 496
 TRichViewEdit 496

EndUndoGroup2 497
 TRichViewEdit 497

EndUpdate 503
 TRichViewEdit 503

EnforceListLevels 426
 TRVHTMLReaderProperties 426

- equation objects 980
- ERichViewError 957
- ErrorCode 796
 - TRVOfficeConverter 796
- EvenColumnsColor 867
 - TRVTableItemInfo 867
- EvenRowsColor 867
 - TRVTableItemInfo 867
- Example
 - Converting table to text 206
 - Dividing table 208
 - Inserting controls what can modify themselves in DBRichViewEdit 625
 - Loading UTF-8 files 460
 - Moving caret to the beginning of paragraph 592
 - OnDropFiles example 593
 - Using viewer-style methods in DBRichViewEdit 624
- ExcludeDocXExportConverter 797
 - TRVOfficeConverter 797
- ExcludeDocXImportConverter 797
 - TRVOfficeConverter 797
- ExcludeHTMLExportConverter 797
 - TRVOfficeConverter 797
- ExcludeHTMLImportConverter 797
 - TRVOfficeConverter 797
- Execute 789, 813
 - TRVDataSourceLink 813
 - TRVSpellChecker 789
- Export of Tables 203
- Export of TRichView contents to HTML 127
- ExportConverters 798
 - TRVOfficeConverter 798
- ExportRTF 800
 - TRVOfficeConverter 800
- ExportRV 800
 - TRVOfficeConverter 800
- Extensions 444
 - TRVMarkdownProperties 444
- ExtensionsInFilter 798
 - TRVOfficeConverter 798
- ExternalCSSFileName 432
 - TRVHTMLSaveProperties 432
- ExtractMetafileBitmaps 454
 - TRVRTFReaderProperties 454
- ExtraStyles 433
 - TRVHTMLSaveProperties 433
- TRVDocParameters 417
- TRVPrint 765
- Field 600, 613
 - TDBRichView 600
 - TDBRichViewEdit 613
- FieldFormat 424, 613
 - TDBRichViewEdit 613
 - TRVDocumentProperty 424
- FieldHighlightColor 637
 - TRVStyle 637
- FieldHighlightType 638
 - TRVStyle 638
- Fill 836
 - TCustomRVPrintPreview 836
- FillLanguageNames 790
 - TRVSpellChecker 790
- Filter 804
 - TRVOfficeConverterInfo 804
- Find 746
 - TRVTabInfos 746
- FindById 691
 - TRVStyleTemplateCollection 691
- FindByName 691
 - TRVStyleTemplateCollection 691
- FindCheckpointByName 287
 - TRichView 287
- FindCheckpointByTag 287
 - TRichView 287
- FindControlItemNo 287
 - TRichView 287
- FindItemId 691
 - TRVStyleTemplateCollection 691
- FindItemByName 691
 - TRVStyleTemplateCollection 691
- FindParaStyle 661
 - TRVStyle 661
- FindStyleWithAlignment 684
 - TParaInfos 684
- FindStyleWithCharset 679
 - TFontInfos 679
- FindStyleWithColor 679
 - TFontInfos 679
- FindStyleWithFont 679
 - TFontInfos 679
- FindStyleWithFontName 680
 - TFontInfos 680
- FindStyleWithFontSize 680
 - TFontInfos 680
- FindStyleWithFontStyle 681
 - TFontInfos 681

- F -

FacingPages 417, 765

- FindStyleWithLevels 687
 - TRVListInfos 687
 - FindSuchStyle 681, 685, 686
 - TFontInfos 681
 - TParaInfos 685
 - TRVListInfos 686
 - FindSuchStyleEx 681, 685
 - TFontInfos 681
 - TParaInfos 685
 - FindTextStyle 661
 - TRVStyle 661
 - Finished 809
 - TRVReportHelper 809
 - FireMonkey 84, 155
 - First 841
 - TCustomRVPrintPreview 841
 - FirstColumnColor 867
 - TRVTableItemInfo 867
 - FirstIndent 722, 751
 - TCustomRVParaInfo 722
 - TRVListLevel 751
 - FirstItemVisible 236
 - TRichView 236
 - FirstJumpNo 236
 - TRichView 236
 - FirstPageNo 823
 - TCustomRVPrint 823
 - FixMarginsMode 766
 - TRVPrint 766
 - FloatingLineColor 638
 - TRVStyle 638
 - Font 751
 - TRVListLevel 751
 - FontName 701, 918
 - TCustomRVFontInfo 701
 - TRVMathItemInfo 918
 - FontQuality 639
 - TRVStyle 639
 - FontSizeDouble 919
 - TRVMathItemInfo 919
 - FontSizePrecision 426
 - TRVHTMLReaderProperties 426
 - FooterY 417, 766
 - TRVDocParameters 417
 - TRVPrint 766
 - FootnoteNumbering 637
 - TRVStyle 637
 - FootnotePageReset 640
 - TRVStyle 640
 - Footnotes 180
 - ForbiddenIds 689
 - TRVStyleTemplateCollection 689
 - ForceFieldHighlight 478
 - TRichViewEdit 478
 - Format 288
 - TRichView 288
 - FormatNextPage 809
 - TRVReportHelper 809
 - FormatPages 774
 - TRVPrint 774
 - FormatString 752, 923
 - TRVListLevel 752
 - TRVSeqItemInfo 923
 - FormatStringW 753
 - TRVListLevel 753
 - FormatTail 288
 - TRichView 288
 - FullRedraw 640
 - TRVStyle 640
- G -**
- GenericFontNames 965
 - TRVDefaultLoadProperties 965
 - GetAllText 983
 - GetAsDifferentUnits 658
 - TRVStyle 658
 - GetAsEMU 658
 - TRVStyle 658
 - GetAsHTMLPixels 658
 - TRVStyle 658
 - GetAsPixels 658
 - TRVStyle 658
 - GetAsRVUnits 658
 - TRVStyle 658
 - GetAsStandardPixels 658
 - TRVStyle 658
 - GetAsTwips 658
 - TRVStyle 658
 - GetAutoCorrection 790
 - TRVSpellChecker 790
 - GetBreakInfo 289
 - TRichView 289
 - GetBulletInfo 289
 - TRichView 289
 - GetCellAt 876
 - TRVTableItemInfo 876
 - GetCheckpointAtCaret 503
 - TRichViewEdit 503
 - GetCheckpointByNo 291

- GetCheckpointByNo 291
 - TRichView 291
- GetCheckpointInfo 291
 - TRichView 291
- GetCheckpointItemNo 291
 - TRichView 291
- GetCheckpointNo 292
 - TRichView 292
- GetCheckpointXY 292
 - TRichView 292
- GetCheckpointY 292
 - TRichView 292
- GetCheckpointYEx 293
 - TRichView 293
- GetControlInfo 293
 - TRichView 293
- GetCurrentBreakInfo 504
 - TRichViewEdit 504
- GetCurrentBulletInfo 505
 - TRichViewEdit 505
- GetCurrentChar 983
- GetCurrentCheckpoint 506
 - TRichViewEdit 506
- GetCurrentControlInfo 506
 - TRichViewEdit 506
- GetCurrentHotspotInfo 507
 - TRichViewEdit 507
- GetCurrentItem 508
 - TRichViewEdit 508
- GetCurrentItemEx 509
 - TRichViewEdit 509
- GetCurrentItemExtraIntProperty 510
- GetCurrentItemExtraIntPropertyEx 510
 - TRichViewEdit 510
- GetCurrentItemExtraStrProperty 510
 - TRichViewEdit 510
- GetCurrentItemExtraStrPropertyEx 510
 - TRichViewEdit 510
- GetCurrentItemText 511
 - TRichViewEdit 511
- GetCurrentItemTextA 511
 - TRichViewEdit 511
- GetCurrentItemTextW 511
 - TRichViewEdit 511
- GetCurrentItemVAlign 512
 - TRichViewEdit 512
- GetCurrentLineCol 512
 - TRichViewEdit 512
- GetCurrentLineText 983
- GetCurrentMisspelling 513
 - TRichViewEdit 513
- GetCurrentParaSectionText 983
- GetCurrentParaText 983
- GetCurrentPictureInfo 513
 - TRichViewEdit 513
- GetCurrentTag 514
 - TRichViewEdit 514
- GetCurrentTextInfo 514
 - TRichViewEdit 514
- GetCurrentWord 983
- GetEditedCell 877
 - TRVTableItemInfo 877
- GetExportFilter 801
 - TRVOfficeConverter 801
- GetFirstCheckpoint 294
 - TRichView 294
- GetFirstItemOnPage 827
 - TCustomRVPrint 827
- GetFocusedItem 295
 - TRichView 295
- GetFooter 978
 - TRVReportHelperWithHeaderFooters 978
- GetFooterRect 775
 - TRVPrint 775
- GetHeader 978
 - TRVReportHelperWithHeaderFooters 978
- GetHeaderRect 775
 - TRVPrint 775
- GetHotspotInfo 295
 - TRichView 295
- GetImportFilter 801
 - TRVOfficeConverter 801
- GetItem 296
 - TRichView 296
- GetItemAt 297
 - TCustomRichView 297
- GetItemCheckpoint 298
 - TRichView 298
- GetItemClientCoords 298
 - TRichView 298
- GetItemCoords 298
 - TRichView 298
- GetItemCoordsEx 299
 - TRichView 299
- GetItemExtraIntProperty 300
 - TRichView 300
- GetItemExtraIntPropertyEx 300
 - TRichView 300
- GetItemExtraStrProperty 300
 - TCustomRichView 300

GetItemExtraStrPropertyEx 300	TCustomRVPrint 828
TCustomRichView 300	GetParaStyleClass 661
GetItemNo 301	TRVStyle 661
TRichView 301	GetPictureInfo 310
GetItemPara 301	TRichView 310
TRichView 301	GetPrevCheckpoint 311
GetItemStyle 302	TRichView 311
TRichView 302	GetRealDocumentPixelsPerInch 311
GetItemTag 302	TRichView 311
TRichView 302	GetRVData 907,960
GetItemText 303	TCustomRVData 960
TRichView 303	TRVTableCellData 907
GetItemTextA 303	GetRVDataText 983
TRichView 303	GetRVESearchOptions 986
GetItemTextW 303	GetRVSearchOptions 987
TRichView 303	GetSelectedImage 312
GetItemVAlign 303	TRichView 312
TRichView 303	GetSelectionBounds 312,878
GetJumpPointItemNo 304	TRichView 312
TRichView 304	TRVTableItemInfo 878
GetJumpPointLocation 304	GetSelText 313
TRichView 304	TRichView 313
GetJumpPointY 305	GetSelTextA 313
TRichView 305	TRichView 313
GetLanguageNameFromCode 790	GetSelTextW 313
TRVSpellChecker 790	TRichView 313
GetLastCheckpoint 305	GetSourceRVData 960
TRichView 305	TCustomRVData 960
GetLastPageHeight 809	GetTextInfo 313
TRVReportHelper 809	TRichView 313
GetLineNo 306	GetTextStyleClass 661
TRichView 306	TRVStyle 661
GetListMarkerInfo 307	GetVisibleText 983
TRichView 307	GetWordAt 314
GetListStyleClass 660	TRichView 314
TRVStyle 660	GetWordAtA 314
GetMainCell 912	TRichView 314
TRVTableRows 912	GetWordAtW 314
GetNextCheckpoint 307	TRichView 314
TRichView 307	GridColor 640
GetNextFileName 375	TRVStyle 640
GetNormalizedSelectionBounds 877	GridReadOnlyColor 640
TRVTableItemInfo 877	TRVStyle 640
GetOffsAfterItem 308	GridReadOnlyStyle 640
TRichView 308	TRVStyle 640
GetOffsBeforeItem 309	GridStyle 640
TRichView 309	TRVStyle 640
GetOriginEx 960	GuessCompletions 791
TCustomRVFormattedData 960	TRVSpellChecker 791
GetPageNo 828	

- H -

- HasNumbering 733, 759
 - TRVListInfo 733
 - TRVListLevel 759
 - HeaderY 418, 767
 - TRVDocParameters 418
 - TRVPrint 767
 - HeadingRowColor 867
 - TRVTableItemInfo 867
 - HeadingRowCount 859
 - TRVTableItemInfo 859
 - Height 938
 - TRVBoxProperties 938
 - HeightType 938
 - TRVBoxProperties 938
 - Hidden text in TRichView 101
 - Hint 590, 901
 - TRVSmartPopupProperties 590
 - TRVTableCellData 901
 - HorizontalAlignment 942
 - TRVBoxPosition 942
 - HorizontalAnchor 941
 - TRVBoxPosition 941
 - HorizontalOffset 942
 - TRVBoxPosition 942
 - HorizontalPositionKind 942
 - TRVBoxPosition 942
 - Hot-Pictures 166
 - Hotspots 172
 - HOutermostRule 859
 - TRVTableItemInfo 859
 - HoverBackColor 702
 - TCustomRVFontInfo 702
 - HoverColor 590, 641, 702
 - TCustomRVFontInfo 702
 - TRVSmartPopupProperties 590
 - TRVStyle 641
 - HoverEffects 703
 - TCustomRVFontInfo 703
 - HoverLineColor 590
 - TRVSmartPopupProperties 590
 - HoverUnderlineColor 703
 - TCustomRVFontInfo 703
 - How to
 - (List) 1062
 - change page size and orientation for printing RichView 1063
 - combine several TRichView documents in one 1068
 - copy one TRichView to another 275
 - create a chat window 1071
 - implement commands like "make bold", "apply font" 1063
 - implement Find and Replace dialogs 1064
 - implement headers and footers 1063
 - implement smart indents 1068
 - make a plain text editor 1066
 - make Unicode editor 1066
 - move a caret to the beginning or to the end of document in TRichViewEdit 1065
 - remove text formatting 1071
 - switch insert/overtyping mode 1064
 - use third-party graphic classes with RichView 1068
 - HRuleColor 860
 - TRVTableItemInfo 860
 - HRuleWidth 860
 - TRVTableItemInfo 860
 - HScrollMax 816
 - TRVScroller 816
 - HScrollPos 816
 - TRVScroller 816
 - HScrollVisible 816
 - TRVScroller 816
 - HTML and TRichView 127
 - HTMLFormatName 1037
 - HTMLPixelsToUnits 659
 - TRVStyle 659
 - HTMLReadProperties 237
 - TRichView 237
 - HTMLSaveProperties 237
 - TRichView 237
 - HTMLSavingType 433
 - TRVHTMLSaveProperties 433
 - HTMLVersion 433
 - TRVHTMLSaveProperties 433
 - Hypertext in TRichView 92
- ## - I -
- Id 734
 - TRVStyleTemplate 734
 - IDAsCheckpoints 427
 - TRVHTMLReaderProperties 427
 - IgnoreBookmarks 454
 - TRVRTFReaderProperties 454
 - IgnoreContentHeight 901
 - TRVTableCellData 901
 - IgnoreEscape 424, 614
 - TDBRichViewEdit 614

IgnoreEscape	424, 614	InsertCheckpoint	516
TRVDocumentProperty	424	TRichViewEdit	516
IgnoreFields	455	InsertCols	878
TRVRTFReaderProperties	455	TRVTableItemInfo	878
IgnoreNotes	455	InsertColsLeft	879
TRVRTFReaderProperties	455	TRVTableItemInfo	879
IgnorePageBreaks	823	InsertColsRight	879
TCustomRVPrint	823	TRVTableItemInfo	879
IgnorePictures	455	InsertControl	517
TRVRTFReaderProperties	455	TRichViewEdit	517
IgnoreSequences	455	InsertDocXFromFileEd	518
TRVRTFReaderProperties	455	TRichViewEdit	518
IgnoreTables	455	InsertDocXFromStreamEd	518
TRVRTFReaderProperties	455	TRichViewEdit	518
ImageHeight	754	InsertFromRVST	691
TRVListLevel	754	TRVStyleTemplateCollection	691
ImageIndex	590, 753	InsertHotPicture	519
TRVListLevel	753	TRichViewEdit	519
TRVSmartPopupProperties	590	InsertHotspot	520
ImageList	590, 754	TRichViewEdit	520
TRVListLevel	754	InsertHTMLFromFileEd	521
TRVSmartPopupProperties	590	TRichViewEdit	521
ImageOptions	434	InsertHTMLFromStreamEd	521
TRVHTMLSaveProperties	434	TRichViewEdit	521
ImagesPrefix	436, 445	InsertItem	522
TRVHTMLSaveProperties	436	TRichViewEdit	522
TRVMarkdownProperties	445	InsertMarkdownFromFileEd	523
ImageVAlign	439	TRichViewEdit	523
TRVMarkdownDefaultItemProperties	439	InsertMarkdownFromStreamEd	524
ImageWidth	754	TRichViewEdit	524
TRVListLevel	754	InsertOEMTextFromFile	525
imgSaveNo	375	TRichViewEdit	525
ImportConverters	798	InsertPageBreak	525
TRVOfficeConverter	798	TRichViewEdit	525
ImportRTF	801	InsertPicture	526
TRVOfficeConverter	801	TRichViewEdit	526
ImportRV	802	InsertRows	880
TRVOfficeConverter	802	TRVTableItemInfo	880
InactiveSelColor	641	InsertRowsAbove	880
TRVStyle	641	TRVTableItemInfo	880
InactiveSelTextColor	641	InsertRowsBelow	880
TRVStyle	641	TRVTableItemInfo	880
Init	809	InsertRTFFromFileEd	527
TRVReportHelper	809	TRichViewEdit	527
InplaceEditor	817	InsertRTFFromStreamEd	528
TRVScroller	817	TRichViewEdit	528
InsertBreak	515	InsertRVFFromFileEd	528
TRichViewEdit	515	TRichViewEdit	528
InsertBullet	515	InsertRVFFromStream	316
TRichViewEdit	515	TRichView	316

- InsertRVFFromStreamEd 529
 - TRichViewEdit 529
 - InsertStringATag 530
 - TRichViewEdit 530
 - InsertStringTag 530
 - TRichViewEdit 530
 - InsertStringWTag 530
 - TRichViewEdit 530
 - InsertTab 531
 - TRichViewEdit 531
 - InsertText 532
 - TRichViewEdit 532
 - InsertTextA 532
 - TRichViewEdit 532
 - InsertTextFromFile 533
 - TRichViewEdit 533
 - InsertTextFromFileW 533
 - TRichViewEdit 533
 - InsertTextW 532
 - TRichViewEdit 532
 - InternalWidth 744
 - TRVBorder 744
 - Intersect 746
 - TRVTabInfos 746
 - InvalidItem 676, 683
 - TFontInfos 676
 - TParaInfos 683
 - InvalidPicture 643
 - TRVStyle 643
 - IsAvailable 791
 - TRVSpellChecker 791
 - IsCellSelected 881
 - TRVTableItemInfo 881
 - IsEqual 711, 717, 727, 731
 - TCustomRVFontInfo 711
 - TCustomRVParaInfo 727
 - TFontInfo 717
 - TParaInfo 731
 - IsFromNewLine 317
 - TRichView 317
 - IsParaStart 318
 - TRichView 318
 - IsValidImporter 803
 - TRVOfficeConverter 803
 - IsWordValid 791
 - TRVSpellChecker 791
 - Item types
 - Breaks 167
 - Bullets 170
 - Controls 168
 - Endnotes 178
 - Footnotes 180
 - Hot-Pictures 166
 - Hotspots 172
 - Labels 176
 - List Markers 174
 - Mathematical expressions 187
 - Numbered sequences 177
 - Page counts 186
 - Page numbers 185
 - Pictures 163
 - References to parent endnotes and footnotes 184
 - Sidenotes 181
 - Tables 173
 - Tabulators 162
 - Text 161
 - Text boxes 183
 - Item types in TRichView 158, 844
 - ItemCount 237
 - TRichView 237
 - Items 677, 683, 686, 689, 745, 749, 803, 910, 912
 - TFontInfos 677
 - TParaInfos 683
 - TRVListInfos 686
 - TRVListLevelCollection 749
 - TRVOfficeCnvList 803
 - TRVStyleTemplateCollection 689
 - TRVTabInfos 745
 - TRVTableRow 910
 - TRVTableRows 912
- J -**
- Jump 714
 - TFontInfo 714
 - JumpCursor 643, 704
 - TCustomRVFontInfo 704
 - TRVStyle 643
- K -**
- KeepTogether 910
 - TRVTableRow 910
 - KeyboardType 478
 - TRichViewEdit 478
 - Keys processed by tables 206
 - Kind 735
 - TRVStyleTemplate 735

- L -

- Labels 176
- Language 787
 - TRVSpellChecker 787
- LanguageIndex 787
 - TRVSpellChecker 787
- Last 842
 - TCustomRVPrintPreview 842
- LastColumnColor 867
 - TRVTableItemInfo 867
- LastItemVisible 238
 - TRichView 238
- LastLineAlignment 719
 - TCustomRVParaInfo 719
- LastPageNo 824
 - TCustomRVPrint 824
- LastRowColor 867
 - TRVTableItemInfo 867
- Lazarus 154
- Leader 747
 - TRVTabInfo 747
- LeftIndent 722, 755
 - TCustomRVParaInfo 722
 - TRVListLevel 755
- LeftMargin 238, 418
 - TRichView 238
 - TRVDocParameters 418
- Levels 732
 - TRVListInfo 732
- LinearToRichView 984
- LineBreaksAsParagraphs 456
 - TRVRTFReaderProperties 456
- LineColor 591
 - TRVSmartPopupProperties 591
- LineCount 238
 - TRichView 238
- LineSelectCursor 643
 - TRVStyle 643
- LineSpacing 722
 - TCustomRVParaInfo 722
- LineSpacingType 723
 - TCustomRVParaInfo 723
- LineWidth 445
 - TRVMarkdownProperties 445
- LineWrapMode 644
 - TRVStyle 644
- ListMarkers 174
- ListLeftIndentPix 439
 - TRVMarkdownDefaultItemProperties 439
- ListMarkerIndentPix 439, 968
 - TRVDefaultLoadProperties 968
 - TRVMarkdownDefaultItemProperties 439
- ListMarkerSavingType 436
 - TRVHTMLSaveProperties 436
- ListStyles 646
 - TRVStyle 646
- ListType 755
 - TRVListLevel 755
- Live spelling check in TRichView 132
- LiveSpellingColor 646
 - TRVStyle 646
- LiveSpellingMode 479
 - TRichViewEdit 479
- LiveSpellingValidateWord 318
 - TRichView 318
- LoadDocX 318
 - TRichView 318
- LoadDocXFromStream 319
 - TRichView 319
- LoadField 604, 622
 - TDBRichView 604
 - TDBRichViewEdit 622
- LoadFromFile 320
 - TRichView 320
- LoadFromFileEx 320
 - TRichView 320
- LoadFromRVST 692
 - TRVStyleTemplateCollection 692
- LoadFromStream 321, 881
 - TRichView 321
 - TRVTableItemInfo 881
- LoadFromStreamEx 321
 - TRichView 321
- LoadFromStreamRVST 692
 - TRVStyleTemplateCollection 692
- LoadHTML 322
 - TRichView 322
- LoadHTMLFromStream 324
 - TRichView 324
- LoadINI 661
 - TRVStyle 661
- LoadMarkdown 325
 - TRichView 325
- LoadMarkdownFromStream 325
 - TRichView 325
- LoadOptions 446
 - TRVMarkdownProperties 446
- LoadReg 662

LoadReg 662
 TRVStyle 662
 LoadRTF 326
 TRichView 326
 LoadRTFFromStream 327
 TRichView 327
 LoadRVF 328
 TRichView 328
 LoadRVFFromStream 329
 TRichView 329
 LoadText 330
 TRichView 330
 LoadTextFromStream 331
 TRichView 331
 LoadTextFromStreamW 331
 TRichView 331
 LoadTextW 330
 TRichView 330

- M -

MainDoc 977
 TRVReportHelperWithHeaderFooters 977
 MainRVStyle 647
 TRVStyle 647
 MakePreview 775
 TRVPrint 775
 MakeScaledPreview 776
 TRVPrint 776
 Margins 768
 TRVPrint 768
 MarginsPen 837
 TCustomRVPrintPreview 837
 MarginsStroke 837
 TCustomRVPrintPreview 837
 MarkdownProperties 239
 TRichView 239
 MarkerAlignment 757
 TRVListLevel 757
 MarkerIndent 757
 TRVListLevel 757
 MarkStylesInUse 331
 TRichView 331
 Mathematical expressions 187
 MaxLength 239
 TRichView 239
 MaxPrintedItemNo 824
 TCustomRVPrint 824
 MaxTextWidth 239
 TRichView 239
 Menu 591
 TRVSmartPopupProperties 591
 MergeCells 881
 TRVTableItemInfo 881
 MergeSelectedCells 882
 TRVTableItemInfo 882
 MetafileCompatibility 824
 TCustomRVPrint 824
 MetafilePixelsPerInch 825
 TCustomRVPrint 825
 MinPrintedItemNo 825
 TCustomRVPrint 825
 MinTextWidth 240
 TRichView 240
 MinWidth 914
 TRVLabelItemInfo 914
 MirrorMargins 418, 769
 TRVDocParameters 418
 TRVPrint 769
 Modified 479
 TRichViewEdit 479
 ModifiedProperties 714, 729
 TFontInfo 714
 TParaInfo 729
 Modifying TRichView items 112
 MoveCaret 534
 TRichViewEdit 534
 MoveRows 882
 TRVTableItemInfo 882

- N -

Name 735, 804
 TRVOfficeConverterInfo 804
 TRVStyleTemplate 735
 Next 736, 842
 TCustomRVPrintPreview 842
 TRVStyleTemplate 736
 NextId 736
 TRVStyleTemplate 736
 NextParaNo 730
 TParaInfo 730
 NextStyleNo 714
 TFontInfo 714
 NoMetafiles 825
 TCustomRVPrint 825
 NormalStyleTemplate 689
 TRVStyleTemplateCollection 689
 NotesSavedAsFootnotes 448
 TRVMarkdownProperties 448

NoteText 240, 927
 TCustomRVNoteltemInfo 927
 TRichView 240

NoVScroll 817
 TRVScroller 817

Numbered sequences 177

NumberType 923, 952
 TCustomRVPageNumberItemInfo 952
 TRVSeqItemInfo 923

- O -

Obtaining items of TRichView 111

OddColumnsColor 867
 TRVTableItemInfo 867

OddRowsColor 867
 TRVTableItemInfo 867

OffsetInCurlItem 479
 TRichViewEdit 479

OnAddStyle 370
 TRichView 370

OnAfterApplyStyle 664
 TRVStyle 664

OnAfterDrawImage 370
 TRichView 370

OnAfterOleDrop 570
 TRichViewEdit 570

OnAfterPrintImage 829
 TCustomRVPrint 829

OnApplyStyle 665
 TRVStyle 665

OnApplyStyleColor 666
 TRVStyle 666

OnAssignImageFileName 371
 TRichView 371

OnBeforeOleDrop 570
 TRichViewEdit 570

OnCaretGetOut 571
 TRichViewEdit 571

OnCaretMove 572
 TRichViewEdit 572

OnCellEditing 892
 TRVTableItemInfo 892

OnChange 572
 TRichViewEdit 572

OnChanging 572
 TRichViewEdit 572

OnCheckpointVisible 372
 TRichView 372

OnCheckStickingItems 573
 TRichViewEdit 573

OnCMHintShow 372
 TRichView 372

OnControlAction 373
 TRichView 373

OnConverting 803
 TRVOfficeConverter 803

OnCopy 374
 TRichView 374

OnCurParaStyleChanged 573
 TRichViewEdit 573

OnCurTextStyleChanged 574
 TRichViewEdit 574

OnDrawBackground 893
 TRVTableItemInfo 893

OnDrawBorder 893
 TRVTableItemInfo 893

OnDrawCheckpoint 667, 830
 TCustomRVPrint 830
 TRVStyle 667

OnDrawCurrentLine 574
 TRichViewEdit 574

OnDrawCustomCaret 575
 TRichViewEdit 575

OnDrawHyperlink 830
 TCustomRVPrint 830

OnDrawPageBreak 668
 TRVStyle 668

OnDrawParaBack 670
 TRVStyle 670

OnDrawStyleText 671
 TRVStyle 671

OnDrawTextBack 673
 TRVStyle 673

OnDropFile 576
 TRichViewEdit 576

OnDropFiles 576
 TRichViewEdit 576

OneLevelPreview 732
 TRVListInfo 732

OnFormatting 780
 TRVPrint 780

OnGetItemCursor 374
 TRichView 374

OnGetPageNumber 811
 TRVReportHelper 811

OnGetSpellingSuggestions 375
 TRichView 375

OnHScrolled 821
 TRVScroller 821

- OnHTMLSaveImage 375
 - TRichView 375
- OnImportFile 378
 - TRichView 378
- OnImportPicture 378
 - TRichView 378
- OnItemAction 380
 - TRichView 380
- OnItemHint 381
 - TRichView 381
- OnItemResize 577
 - TRichViewEdit 577
- OnItemTextEdit 578
 - TRichViewEdit 578
- OnJump 382
 - TRichView 382
- OnLoadCustomFormat 383
 - TRichView 383
- OnLoadDocument 384
 - TRichView 384
- OnMeasureCustomCaret 578
 - TRichViewEdit 578
- OnNewDocument 384
 - TRichView 384
- OnOleDragEnter 579
 - TRichViewEdit 579
- OnOleDragLeave 579
 - TRichViewEdit 579
- OnOleDragOver 580
 - TRichViewEdit 580
- OnOleDrop 581
 - TRichViewEdit 581
- OnPagePostpaint 781
 - TRVPrint 781
- OnPagePrepaint 782
 - TRVPrint 782
- OnPaint 384
 - TRichView 384
- OnParaStyleConversion 582
 - TRichViewEdit 582
- OnPaste 584
 - TRichViewEdit 584
- OnPrintComponent 831
 - TCustomRVPrint 831
- OnProgress 385
 - TRichView 385
- OnReadField 386, 390
 - TCustomRichView 386, 390
- OnReadHyperlink 388
 - TCustomRichView 388
- OnRVDbClick 391
 - TRichView 391
- OnRVFControlNeeded 392
 - TRichView 392
- OnRVFImageListNeeded 393
 - TRichView 393
- OnRVFPictureNeeded 393
 - TRichView 393
- OnRVMouseDown 394
 - TRichView 394
- OnRVMouseMove 395
 - TRichView 395
- OnRVMouseUp 396
 - TRichView 396
- OnRVRightClick 397
 - TRichView 397
- OnSaveComponentToFile 397
 - TRichView 397
- OnSaveCustomFormat 585
 - TRichViewEdit 585
- OnSaveDocXExtra 399
 - TRichView 399
- OnSaveHTMLExtra 401
 - TRichView 401
- OnSaveImage2 402
 - TRichView 402
- OnSaveItemToFile 404
 - TRichView 404
- OnSaveParaToHTML 406
 - TRichView 406
- OnSaveRTFExtra 406
 - TRichView 406
- OnSelect 408
 - TRichView 408
- OnSendingToPrinter 782
 - TRVPrint 782
- OnSmartPopupClick 585
 - TRichViewEdit 585
- OnSpellForm 793
 - TRVSpellChecker 793
- OnSpellFormAction 794
 - TRVSpellChecker 794
- OnSpellFormLocalize 795
 - TRVSpellChecker 795
- OnSpellingCheck 409
 - TRichView 409
- OnStyleConversion 585
 - TRichViewEdit 585
- OnStyleHoverSensitive 674
 - TRVStyle 674

-
- OnStyleTemplatesChange 410
 - TRichView 410
 - OnTextFound 410
 - TRichView 410
 - OnVScrolled 821
 - TRVScroller 821
 - OnWriteHyperlink 411
 - TRichView 411
 - OnWriteObjectProperties 413
 - TRichView 413
 - OnZoomChanged 843
 - TCustomRVPrintPreview 843
 - Opacity 742, 860, 902
 - TRVBackgroundRect 742
 - TRVTableCellData 902
 - TRVTableItemInfo 860
 - Options 240, 704, 723, 758, 861, 902
 - TCustomRVFontInfo 704
 - TCustomRVParaInfo 723
 - TRichView 240
 - TRVListLevel 758
 - TRVTableCellData 902
 - TRVTableItemInfo 861
 - Orientation 418
 - TRVDocParameters 418
 - OutlineLevel 725
 - TCustomRVParaInfo 725
 - Overview 84
 - Overview of item types 158
 - Overview of tables 189
- P -**
- Page counts 186
 - Page numbers 185
 - PageBorderColor 837
 - TCustomRVPrintPreview 837
 - PageBorderWidth 838
 - TCustomRVPrintPreview 838
 - PageBreakBefore 910
 - TRVTableRow 910
 - PageBreakColor 647
 - TRVStyle 647
 - PageBreaksBeforeItems 244
 - TRichView 244
 - PageHeight 419, 784
 - TRVDocParameters 419
 - TRVVirtualPrinterProperties 784
 - PageNo 838
 - TCustomRVPrintPreview 838
 - PageNoFromNumber 826
 - TCustomRVPrint 826
 - PagesCount 826
 - TCustomRVPrint 826
 - PageWidth 419, 784
 - TRVDocParameters 419
 - TRVVirtualPrinterProperties 784
 - Paragraphs in TRichView 95
 - ParaStyle 737
 - TRVStyleTemplate 737
 - ParaStyleMode 456
 - TRVRTFReaderProperties 456
 - ParaStyleNo 457
 - TRVRTFReaderProperties 457
 - ParaStyles 648
 - TRVStyle 648
 - ParaStyleTemplateId 705
 - TCustomRVFontInfo 705
 - Parent 737
 - TRVStyleTemplate 737
 - ParentId 737
 - TRVStyleTemplate 737
 - Paste 534
 - TRichViewEdit 534
 - PasteBitmap 535
 - TRichViewEdit 535
 - PasteGraphicFile 535
 - TRichViewEdit 535
 - PasteHTML 536
 - TRichViewEdit 536
 - PasteMetafile 537
 - TRichViewEdit 537
 - PasteRTF 537
 - TRichViewEdit 537
 - PasteRVF 538
 - TRichViewEdit 538
 - PasteText 538
 - TRichViewEdit 538
 - PasteTextA 538
 - TRichViewEdit 538
 - PasteTextW 538
 - TRichViewEdit 538
 - PasteURL 539
 - TRichViewEdit 539
 - Path 804
 - TRVOfficeConverterInfo 804
 - Picture 759
 - TRVListLevel 759
 - Pictures 163
 - PixelsPerInch 677
-

PixelsPerInch 677
 TFontInfos 677
 PixelsToUnits 659
 TRVStyle 659
 Popup 591
 TRVSmartPopupProperties 591
 Position 591, 748
 TRVSmartPopupProperties 591
 TRVTabInfo 748
 PositionInText 944
 TRVBoxPosition 944
 Prev 842
 TCustomRVPrintPreview 842
 Preview100PercentHeight 769
 TRVPrint 769
 Preview100PercentWidth 769
 TRVPrint 769
 PreviewCorrection 826
 TCustomRVPrint 826
 PreviewMode 799
 TRVOfficeConverter 799
 Print 776
 TRVPrint 776
 PrintableAreaPen 838
 TCustomRVPrintPreview 838
 PrintableAreaStroke 838
 TCustomRVPrintPreview 838
 PrintOptions 863
 TRVTableItemInfo 863
 PrintPages 777
 TRVPrint 777
 Protection 705
 TCustomRVFontInfo 705
 ProtectTextStyleNo 914
 TRVLabelItemInfo 914
 PRVPDFMetadata 1018
 PRVScreenAndDevice 1024

- Q -

QuickAccess 737
 TRVStyleTemplate 737

- R -

ReadBackground 427
 TRVHTMLReaderProperties 427
 ReadDocParameters 427, 457
 TRVHTMLReaderProperties 427
 TRVRTFReaderProperties 457

ReadNoteNumberingType 457
 TRVRTFReaderProperties 457
 ReadOnly 480, 614
 TDBRichViewEdit 614
 TRichViewEdit 480
 Redo 540
 TRichViewEdit 540
 RedoAction 540
 TRichViewEdit 540
 RedoName 540
 TRichViewEdit 540
 References to parent endnotes and footnotes 184
 Reformat 332
 TRichView 332
 RefreshListMarkers 332
 TRichView 332
 RefreshSequences 332
 TRichView 332
 RegisterAsService 791
 TRVSpellChecker 791
 RegisterEditor 792
 TRVSpellChecker 792
 RelativeToCell 945
 TRVBoxPosition 945
 RemoveCheckpoint 332
 TRichView 332
 RemoveCheckpointAtCaret 541
 TRichViewEdit 541
 RemoveCheckpointEd 541
 TRichViewEdit 541
 RemoveCurrentCheckpoint 541
 TRichViewEdit 541
 RemoveCurrentPageBreak 542
 TRichViewEdit 542
 RemoveInternalLeading 915
 TRVLabelItemInfo 915
 RemoveLists 542
 TRichViewEdit 542
 RemoveSpaceBelow 915
 TRVLabelItemInfo 915
 ReplaceDocumentEd 927
 TCustomRVNotelItemInfo 927
 Reset 421, 923, 978
 TRVDocParameters 421
 TRVReportHelperWithHeaderFooters 978
 TRVSeqItemInfo 923
 ResetAnimation 333
 TRichView 333
 ResetLayout 421
 TRVDocParameters 421

-
- ResetNameCounter 693
 - TRVStyleTemplateCollection 693
 - ResizeControl 542
 - TRichViewEdit 542
 - ResizeCurrentControl 543
 - TRichViewEdit 543
 - RichView 32, 807
 - TRVReportHelper 807
 - RichView Format (RVF) overview 124
 - RichViewAllowCopyTableCells 1042
 - RichViewAlternativePicPrint 1043
 - RichViewApostropheInWord 1043
 - RichViewAutoAssignNormalStyleTemplate 1043
 - RichViewCompareStyleNames 1044
 - RichViewCSSUseSystemColors 1044
 - RichViewCtrlUpDownScroll 1044
 - RichViewDoNotCheckRVFStyleRefs 1045
 - RichViewDoNotMergeNumbering 1045
 - RichViewEditCaretHeightExtra 1040
 - RichViewEditCaretPosition 1040
 - RichViewEditCaretWidth 1040
 - RichViewEditCaretWidthMode 1040
 - RichViewEditCustomCaretSize 1040
 - RichViewEditDefaultProportionalResize 1045
 - RichViewEditEnterAllowsEmptyMarkeredLines 1046
 - RichViewEditMaxCaretHeight 1040
 - RichViewEditMinCaretHeight 1040
 - RichViewJustifyBeforeLineBreak 1046
 - RichViewMaxPictureCount 1046
 - RichViewMaxThumbnailCount 1047
 - RichViewPixelsPerInch 784, 1047
 - RichViewResetStandardFlag 1047
 - RichViewSaveDivInHTMLEx 1040
 - RichViewSaveHyperlinksInDocXAsFields 1048
 - RichViewSavePageBreaksInText 1040
 - RichViewSavePIInHTML 1040
 - RichViewShowGhostSpaces 1048
 - RichViewStringFormatMethod 1048
 - RichViewTableAutoAddRow 1049
 - RichViewTableDefaultRTFAutoFit 1049
 - RichViewToLinear 984
 - RichViewUnicodeInput 1050
 - RightIndent 725
 - TCustomRVParaInfo 725
 - RightMargin 244, 419
 - TRichView 244
 - TRVDocParameters 419
 - Rotation 904
 - TRVTableCellData 904
 - RowBandSize 858
 - TRVTableItemInfo 858
 - RowCount 864
 - TRVTableItemInfo 864
 - Rows 864
 - TRVTableItemInfo 864
 - RowSpan 905
 - TRVTableCellData 905
 - RTFErrorCode 457
 - TRVRTFReaderProperties 457
 - RTFOptions 245
 - TRichView 245
 - RTFormatName 1037
 - RTFReadProperties 246
 - TRichView 246
 - Rules of tables 193
 - RV_AfterImportGraphic 1068
 - RV_CP_DEFAULT 1050
 - RV_CreateGraphics 1068
 - RV_EMUtoUnits 981, 987
 - RV_GetPrinterDC 988
 - RV_RegisterHTMLGraphicFormat 1068
 - RV_RegisterPngGraphic 1068
 - RV_TestFileUnicode 992
 - RV_TestStreamUnicode 992
 - RV_TestStringUnicode 992
 - RV_TwipsToUnits 981, 987
 - RV_UnitsPerInch 981, 987
 - RV_UnitsToEMU 981, 987
 - RV_UnitsToPixels 981, 987
 - RV_UnitsToTwips 981, 987
 - RV_UnitsToUnits 981, 987
 - RVAllFontInfoProperties 1008
 - RVAllParaInfoProperties 1017
 - RVAlwaysShowPlaceholders 1051
 - RVApplyMathProperties 986
 - RVCharsPerLineBreak 984
 - rvclNone 702
 - RVControlsPainter 1051
 - RVData 247
 - TRichView 247
 - RVDefaultLoadProperties 1051
 - rvdoALL 999
 - rveipcAlignment 1051, 1056
 - rveipcBreakColor 1051
 - rveipcBreakStyle 1051
 - rveipcBreakWidth 1051
 - rveipcCursor 1051, 1056
 - rveipcDisplayInline 1051

- rveipcFontSizeDouble 1051
 - rveipcHotImageIndex 1051
 - rveipcMinWidth 1051, 1056
 - rveipcPercentWidth 1051
 - rveipcProtectTextStyleNo 1051, 1056
 - rveipcRemoveInternalLeading 1051, 1056
 - rveipcSeqNumberType 1051, 1056
 - rveipcSeqReset 1051, 1056
 - rveipcSeqStartFrom 1051, 1056
 - rveipcTextColor 1051
 - RVEMPTYTAG 1056
 - RVF 124, 143, 230
 - RVF Specification 143
 - RVFOptions 247
 - TRichView 247
 - RVFormatName 1037
 - RVFParaStylesReadMode 251
 - TRichView 251
 - RVFTextStylesReadMode 251
 - TRichView 251
 - RVFWarnings 251
 - TRichView 251
 - RVGetFirstEndnote 989
 - RVGetFirstFootnote 989
 - RVGetFirstSidenote 989
 - RVGetLinearCaretPos 984
 - RVGetMathProperties 986
 - RVGetNextEndnote 989
 - RVGetNextFootnote 989
 - RVGetNextSidenote 989
 - RVGetNoteTextStyleNo 990
 - RVGetSelection 984
 - RVGetSelectionEx 984
 - RVGetSystemColor 1057
 - RVGetTextLength 984
 - RVGetTextRange 984
 - RVGraphicHandler 1057
 - RVIsCustomURL 1058
 - RVIsEmail function 991
 - RVIsURL function 991
 - RVMathEquationManager function 991
 - RVNonTextCharacter 984
 - RVOpenURL procedure 991
 - RVParagraphMarks 1058
 - RVPrint 628
 - TRVPrintPreview 628
 - rvsBreak 1059
 - rvsBullet 1059
 - rvsComponent 1059
 - rvsDefStyle 1059
 - rvsEndnote 1059
 - RVSetLinearCaretPos 984
 - RVSetMathProperties 986
 - RVSetSelection 984
 - RVSetSelectionEx 984
 - rvsFootnote 1059
 - rvsHeading 1059
 - rvsHotPicture 1059
 - rvsHotspot 1059
 - rvsJump1 1059
 - rvsJump2 1059
 - rvsKeyword 1059
 - rvsLabel 1059
 - rvsListMarker 1059
 - rvsMathEquation 1059
 - rvsNormal 1059
 - rvsNoteReference 1059
 - rvsPicture 1059
 - rvsSequence 1059
 - rvsSubheading 1059
 - rvsTab 1059
 - rvsTable 1059
 - RVStyle.TRVColorMode 997
 - RVThumbnailMaker 1061
 - RVU_AnsiToUnicode 992
 - RVU_AnsiToUTF8 992
 - RVU_UnicodeToAnsi 992
 - RVUnitsToUnits 659
 - TRVStyle 659
 - RVVisibleSpecialCharacters 1061
- S -**
- SaveCSS 662
 - TRVStyle 662
 - SaveCSSToStream 663
 - TRVStyle 663
 - SaveDocX 333
 - TRichView 333
 - SaveDocXToStream 334
 - TRichView 334
 - SaveHeaderAndFooter 437
 - TRVHTMLSaveProperties 437
 - SaveHiddenText 448
 - TRVMarkdownProperties 448
 - SaveHTML 335, 336
 - TRichView 335, 336
 - SaveHTMLEx 337

- SaveHTMLEx 337
 - TRichView 337
- SaveHTMLToStream 338, 340
 - TRichView 338, 340
- SaveHTMLToStreamEx 340
 - TRichView 340
- SaveINI 663
 - TRVStyle 663
- SaveMarkdown 340
 - TRichView 340
- SaveMarkdownToStream 341
 - TRichView 341
- SaveMathMLInHTML 969
- SaveOMMLInDocX 969
- SaveOptions 448
 - TRVMarkdownProperties 448
- SavePDF 778, 810
 - TRVPrint 778
 - TRVReportHelper 810
- SavePicture 342
 - TRichView 342
- SaveReg 664
 - TRVStyle 664
- SaveRowsToStream 883
 - TRVTableItemInfo 883
- SaveRTF 343
 - TRichView 343
- SaveRTFToStream 344
 - TRichView 344
- SaveRVF 344
 - TRichView 344
- SaveRVFToStream 344
 - TRichView 344
- SaveText 345
 - TRichView 345
- SaveTextToStream 345
 - TRichView 345
- SaveTextToStreamW 345
 - TRichView 345
- SaveTextW 345
 - TRichView 345
- SaveToRVST 692
 - TRVStyleTemplateCollection 692
- SaveToStream 883
 - TRVTableItemInfo 883
- SaveToStreamRVST 692
 - TRVStyleTemplateCollection 692
- Scrolling in TRichView 105
- ScrollTo 820
 - TRVScroller 820
- Searching and replacing in TRichView and TRichViewEdit 110
- SearchText 346, 543
 - TRichView 346
 - TRichViewEdit 543
- SearchTextA 346, 543
 - TRichView 346
 - TRichViewEdit 543
- SearchTextW 346, 543
 - TRichView 346
 - TRichViewEdit 543
- SelColor 641
 - TRVStyle 641
- Select 884
 - TRVTableItemInfo 884
- SelectAll 348
 - TRichView 348
- SelectCols 884
 - TRVTableItemInfo 884
- SelectControl 348
 - TRichView 348
- SelectCurrentLine 545
 - TRichViewEdit 545
- SelectCurrentWord 545
 - TRichViewEdit 545
- Selecting parts of TRichView document 107
- Selection in Table 195
- SelectionExists 349
 - TRichView 349
- SelectionHandleKind 648
 - TRVStyle 648
- SelectionHandlesVisible 253
 - TRichView 253
- SelectionMode 649
 - TRVStyle 649
- SelectionStyle 649
 - TRVStyle 649
- SelectRows 884
 - TRVTableItemInfo 884
- SelectWordAt 349
 - TRichView 349
- SelOpacity 641
 - TRVStyle 641
- SelTextColor 641
 - TRVStyle 641
- Semitransparent Objects in TRichView 121
- SeqName 924
 - TRVSeqItemInfo 924
- SetAddParagraphMode 350
 - TRichView 350

- SetBackgroundImageEd 546
TRichViewEdit 546
- SetBreakInfo 351
TRichView 351
- SetBreakInfoEd 547
TRichViewEdit 547
- SetBulletInfo 352
TRichView 352
- SetBulletInfoEd 548
TRichViewEdit 548
- SetButtonState 592
TRVSmartPopupProperties 592
- SetCellBackgroundImage 885
TRVTableItemInfo 885
- SetCellBackgroundImageFileName 885
TRVTableItemInfo 885
- SetCellBackgroundStyle 885
TRVTableItemInfo 885
- SetCellBestHeight 886
TRVTableItemInfo 886
- SetCellBestWidth 886
TRVTableItemInfo 886
- SetCellBorderColor 886
TRVTableItemInfo 886
- SetCellBorderLightColor 886
TRVTableItemInfo 886
- SetCellColor 887
TRVTableItemInfo 887
- SetCellHint 887
TRVTableItemInfo 887
- SetCellOpacity 887
TRVTableItemInfo 887
- SetCellOptions 887
TRVTableItemInfo 887
- SetCellRotation 888
TRVTableItemInfo 888
- SetCellTag 888
TRVTableItemInfo 888
- SetCellVAlign 888
TRVTableItemInfo 888
- SetCellVisibleBorders 889
TRVTableItemInfo 889
- SetCheckpointInfo 353
TRichView 353
- SetCheckpointInfoEd 549
TRichViewEdit 549
- SetControlInfo 354
TRichView 354
- SetControlInfoEd 550
TRichViewEdit 550
- SetCurrentBreakInfo 551
TRichViewEdit 551
- SetCurrentBulletInfo 552
TRichViewEdit 552
- SetCurrentCheckpointInfo 553
TRichViewEdit 553
- SetCurrentControlInfo 554
TRichViewEdit 554
- SetCurrentHotspotInfo 555
TRichViewEdit 555
- SetCurrentItemExtraIntProperty 556
TRichViewEdit 556
- SetCurrentItemExtraIntPropertyEx 556
TRichViewEdit 556
- SetCurrentItemExtraStrProperty 557
TRichViewEdit 557
- SetCurrentItemExtraStrPropertyEx 557
TRichViewEdit 557
- SetCurrentItemText 557
TRichViewEdit 557
- SetCurrentItemTextA 557
TRichViewEdit 557
- SetCurrentItemTextW 557
TRichViewEdit 557
- SetCurrentItemVAlign 558
TRichViewEdit 558
- SetCurrentPictureInfo 559
TRichViewEdit 559
- SetCurrentTag 560
TRichViewEdit 560
- SetFloatPropertyEd 545
TRichViewEdit 545
- SetFooter 355, 778
TRichView 355
TRVPrint 778
- SetHeader 356, 779
TRichView 356
TRVPrint 779
- SetHotspotInfo 357
TRichView 357
- SetHotspotInfoEd 560
TRichViewEdit 560
- SetIntPropertyEd 545
TRichViewEdit 545
- SetItemExtraIntProperty 358
TRichView 358
- SetItemExtraIntPropertyEd 562
TRichViewEdit 562
- SetItemExtraIntPropertyEx 358
TRichView 358

SetItemExtraIntPropertyExEd 562	SetZoom 843
TRichViewEdit 562	TCustomRVPrintPreview 843
SetItemExtraStrProperty 359	ShadowColor 839
TRichView 359	TCustomRVPrintPreview 839
SetItemExtraStrPropertyEd 562	ShadowWidth 839
TRichViewEdit 562	TCustomRVPrintPreview 839
SetItemExtraStrPropertyEx 359	ShortCut 591
TRichView 359	TRVSmartPopupProperties 591
SetItemExtraStrPropertyExEd 562	SidenoteNumbering 637
TRichViewEdit 562	TRVStyle 637
SetItemTag 360	Sidenotes 181
TRichView 360	SingleClick 253
SetItemTagEd 563	TRichView 253
TRichViewEdit 563	SingleSymbols 705
SetItemText 360	TCustomRVFontInfo 705
TRichView 360	Size 707
SetItemTextA 360	TCustomRVFontInfo 707
TRichView 360	SizeDouble 707
SetItemTextEd 564	TCustomRVFontInfo 707
TRichViewEdit 564	Skia 157
SetItemTextEdA 564	Skia4Delphi 157
TRichViewEdit 564	SkipHiddenText 428, 458
SetItemTextEdW 564	TRVHTMLReaderProperties 428
TRichViewEdit 564	TRVRTFReaderProperties 458
SetItemTextW 360	Smart Popups 588
TRichView 360	SmartPopupProperties 480
SetItemVAlign 361	TRichViewEdit 480
TRichView 361	SmartPopupVisible 481
SetItemVAlignEd 565	TRichViewEdit 481
TRichViewEdit 565	Smooth image scaling in TRichView 120
SetListMarkerInfo 362	SoftPageBreakColor 651
TRichView 362	TRVStyle 651
SetPictureInfo 363	SpaceAfter 726
TRichView 363	TCustomRVParaInfo 726
SetPictureInfoEd 565	SpaceBefore 726
TRichViewEdit 565	TCustomRVParaInfo 726
SetRowKeepTogether 889	SpacesInTab 652
TRVTableItemInfo 889	TRVStyle 652
SetRowPageBreakBefore 889	SpecialCharactersColor 652
TRVTableItemInfo 889	TRVStyle 652
SetRowVAlign 889	SpellFormStyle 788
TRVTableItemInfo 889	TRVSpellChecker 788
SetSelectionBounds 365	SplitSelectedCellsHorizontally 890
TRichView 365	TRVTableItemInfo 890
SetStrPropertyEd 545	SplitSelectedCellsVertically 890
TRichViewEdit 545	TRVTableItemInfo 890
SetTableVisibleBorders 889	Standard 695
TRVTableItemInfo 889	TCustomRVInfo 695
SetUndoGroupMode 567	StandardPixelsToUnits 659
TRichViewEdit 567	TRVStyle 659

- StartAnimation 366
 - TRichView 366
 - StartAt 826
 - TCustomRVPrint 826
 - StartFrom 759, 924
 - TRVListLevel 759
 - TRVSeqItemInfo 924
 - StartLiveSpelling 367, 792
 - TRichView 367
 - TRVSpellChecker 792
 - StopAnimation 367
 - TRichView 367
 - StoreSearchResult 367
 - TRichView 367
 - Stream 799
 - TRVOfficeConverter 799
 - Style 253, 708, 744
 - TCustomRVFontInfo 708
 - TRichView 253
 - TRVBorder 744
 - style templates 729
 - StyleEx 708
 - TCustomRVFontInfo 708
 - StyleName 696
 - TCustomRVInfo 696
 - Styles and Style-Templates in TRichView 98
 - StyleTemplateId 696
 - TCustomRVFontOrParaInfo 696
 - StyleTemplateInsertMode 254
 - TRichView 254
 - StyleTemplates 652
 - TRVStyle 652
 - SubSuperScriptType 708
 - TCustomRVFontInfo 708
 - Suggest 792
 - TRVSpellChecker 792
 - SupportsFeatures 793
 - TRVSpellChecker 793
- T -**
- Table Cells 192
 - Table Operations 199
 - Table Resizing 205
 - TableCellWidthPix 439
 - TRVMarkdownDefaultItemProperties 439
 - Tables 173
 - Tables and Compatibility 191
 - Tables in RichView 190
 - Tables in TRichView 189
 - TabNavigation 255
 - TRichView 255
 - Tabs 726
 - TCustomRVParaInfo 726
 - Tabulators 162
 - Tag 906
 - TRVTableCellData 906
 - Tags of TRichView items 91
 - TargetPixelsPerInch 806
 - TRVReportHelper 806
 - TBackgroundStyle 223
 - TCheckpointData 993
 - TConvertingEvent 803
 - TCPEventKind 227
 - TCustomRichView 210, 393
 - TCustomRichView.AssignSoftPageBreaks 277
 - TCustomRichView.GetItemAt 297
 - TCustomRichView.GetItemExtraStrProperty 300
 - TCustomRichView.GetItemExtraStrPropertyEx 300
 - TCustomRichView.OnReadField 386, 390
 - TCustomRichView.OnReadHyperlink 388
 - TCustomRichViewEdit 461, 986
 - TCustomRichViewEdit.ActualCurTextStyleNo 472
 - TCustomRVControl 813
 - TCustomRVData 957
 - TCustomRVData.Edit 959
 - TCustomRVData.GetRVData 960
 - TCustomRVData.GetSourceRVData 960
 - TCustomRVFieldItemInfo 952
 - TCustomRVFontInfo 696
 - TCustomRVFontInfo.Assign 711
 - TCustomRVFontInfo.AssignTo 711
 - TCustomRVFontInfo.BackColor 698
 - TCustomRVFontInfo.BiDiMode 699
 - TCustomRVFontInfo.CharScale 699
 - TCustomRVFontInfo.Charset 700
 - TCustomRVFontInfo.CharSpacing 700
 - TCustomRVFontInfo.Color 701
 - TCustomRVFontInfo.Create 711
 - TCustomRVFontInfo.EmptyWidth 701
 - TCustomRVFontInfo.FontName 701
 - TCustomRVFontInfo HoverBackColor 702
 - TCustomRVFontInfo HoverColor 702
 - TCustomRVFontInfo HoverEffects 703
 - TCustomRVFontInfo HoverUnderlineColor 703
 - TCustomRVFontInfo.IsEqual 711
 - TCustomRVFontInfo.JumpCursor 704
 - TCustomRVFontInfo.Options 704
 - TCustomRVFontInfo.ParaStyleTemplateId 705

TCustomRVFontInfo.Protection	705	TCustomRVPrint.EndAt	823
TCustomRVFontInfo.SingleSymbols	705	TCustomRVPrint.FirstPageNo	823
TCustomRVFontInfo.Size	707	TCustomRVPrint.GetFirstItemOnPage	827
TCustomRVFontInfo.SizeDouble	707	TCustomRVPrint.GetPageNo	828
TCustomRVFontInfo.Style	708	TCustomRVPrint.IgnorePageBreaks	823
TCustomRVFontInfo.StyleEx	708	TCustomRVPrint.LastPageNo	824
TCustomRVFontInfo.SubSuperScriptType	708	TCustomRVPrint.MaxPrintedItemNo	824
TCustomRVFontInfo.UnderlineColor	709	TCustomRVPrint.MetafileCompatibility	824
TCustomRVFontInfo.UnderlineType	709	TCustomRVPrint.MetafilePixelsPerInch	825
TCustomRVFontInfo.VShift	710	TCustomRVPrint.MinPrintedItemNo	825
TCustomRVFontOrParaInfo.StyleTemplateld	696	TCustomRVPrint.NoMetafiles	825
TCustomRVFormattedData	957	TCustomRVPrint.OnAfterPrintImage	829
TCustomRVFormattedData.GetOriginEx	960	TCustomRVPrint.OnDrawCheckpoint	830
TCustomRVInfo	693	TCustomRVPrint.OnDrawHyperlink	830
TCustomRVInfo.Standard	695	TCustomRVPrint.OnPrintComponent	831
TCustomRVInfo.StyleName	696	TCustomRVPrint.PageNoFromNumber	826
TCustomRVIsURLFunction	1058	TCustomRVPrint.PagesCount	826
TCustomRVItemInfo	844	TCustomRVPrint.PreviewCorrection	826
TCustomRVNoteltemInfo	844, 925	TCustomRVPrint.StartAt	826
TCustomRVNoteltemInfo.CreateEx	927	TCustomRVPrint.TransparentBackground	827
TCustomRVNoteltemInfo.Document	926	TCustomRVPrint.UpdatePaletteInfo	828
TCustomRVNoteltemInfo.NoteText	927	TCustomRVPrintPreview	832
TCustomRVNoteltemInfo.ReplaceDocumentEd	927	TCustomRVPrintPreview.ClickMode	836
TCustomRVPageNumberItemInfo	952	TCustomRVPrintPreview.Color	836
TCustomRVPageNumberItemInfo.NumberType	952	TCustomRVPrintPreview.Create	841
TCustomRVParaInfo	718	TCustomRVPrintPreview.DarkModeUI	837
TCustomRVParaInfo.Alignment	719	TCustomRVPrintPreview.Fill	836
TCustomRVParaInfo.Assign	726	TCustomRVPrintPreview.First	841
TCustomRVParaInfo.Background	721	TCustomRVPrintPreview.Last	842
TCustomRVParaInfo.BiDiMode	721	TCustomRVPrintPreview.MarginsPen	837
TCustomRVParaInfo.Border	721	TCustomRVPrintPreview.MarginsStroke	837
TCustomRVParaInfo.Create	727	TCustomRVPrintPreview.Next	842
TCustomRVParaInfo.FirstIndent	722	TCustomRVPrintPreview.OnZoomChanged	843
TCustomRVParaInfo.IsEqual	727	TCustomRVPrintPreview.PageBorderColor	837
TCustomRVParaInfo.LastLineAlignment	719	TCustomRVPrintPreview.PageBorderWidth	838
TCustomRVParaInfo.LeftIndent	722	TCustomRVPrintPreview.PageNo	838
TCustomRVParaInfo.LineSpacing	722	TCustomRVPrintPreview.Prev	842
TCustomRVParaInfo.LineSpacingType	723	TCustomRVPrintPreview.PrintableAreaPen	838
TCustomRVParaInfo.Options	723	TCustomRVPrintPreview.PrintableAreaStroke	838
TCustomRVParaInfo.OutlineLevel	725	TCustomRVPrintPreview.SetZoom	843
TCustomRVParaInfo.RightIndent	725	TCustomRVPrintPreview.ShadowColor	839
TCustomRVParaInfo.SpaceAfter	726	TCustomRVPrintPreview.ShadowWidth	839
TCustomRVParaInfo.SpaceBefore	726	TCustomRVPrintPreview.ZoomInCursor	839
TCustomRVParaInfo.Tabs	726	TCustomRVPrintPreview.ZoomMode	840
TCustomRVPrint	822	TCustomRVPrintPreview.ZoomOutCursor	840
TCustomRVPrint.CanSavePDF	827	TCustomRVPrintPreview.ZoomPercent	841
TCustomRVPrint.Clear	827	TDBRichView	594
TCustomRVPrint.ColorMode	822	TDBRichView.AllowMarkdown	598
TCustomRVPrint.DarkMode	823	TDBRichView.AutoDeleteUnusedStyles	599

- TDBRichView.AutoDisplay 599
- TDBRichView.CodePage 599
- TDBRichView.Create 604
- TDBRichView.DataField 600
- TDBRichView.DataSource 600
- TDBRichView.Destroy 604
- TDBRichView.Field 600
- TDBRichView.LoadField 604
- TDBRichViewEdit 606
 - Inserting controls what can modify themselves in DBRichViewEdit 625
 - Using viewer-style methods in DBRichViewEdit 624
- TDBRichViewEdit.AllowMarkdown 611
- TDBRichViewEdit.AutoDeleteUnusedStyles 611
- TDBRichViewEdit.AutoDisplay 612
- TDBRichViewEdit.Change 621
- TDBRichViewEdit.CodePage 612
- TDBRichViewEdit.Create 621
- TDBRichViewEdit.DataField 613
- TDBRichViewEdit.DataSource 613
- TDBRichViewEdit.Destroy 621
- TDBRichViewEdit.Field 613
- TDBRichViewEdit.FieldFormat 613
- TDBRichViewEdit.IgnoreEscape 614
- TDBRichViewEdit.LoadField 622
- TDBRichViewEdit.ReadOnly 614
- TDBRichViewEdit.TRVDocumentProperty 424
- Text 915,919
 - TRVLabelItemInfo 915
 - TRVMathItemInfo 919
- Text boxes 183
- Text Items 161
- TextBackgroundKind 653
 - TRVStyle 653
- TextColor 920
 - TRVMathItemInfo 920
- TextColSeparator 865
 - TRVTableItemInfo 865
- TextEngine 654
 - TRVStyle 654
- TextRowSeparator 865
 - TRVTableItemInfo 865
- TextStyle 737
 - TRVStyleTemplate 737
- TextStyleMode 458
 - TRVRTFReaderProperties 458
- TextStyleNo 459,915
 - TRVLabelItemInfo 915
 - TRVRTFReaderProperties 459
- TextStyles 654
- TRVStyle 654
- TFontInfo 712
 - Assign 715
 - Create 716
 - Draw 716
 - IsEqual 717
 - Jump 714
 - ModifiedProperties 714
 - NextStyleNo 714
 - Unicode 715
- TFontInfos 675
 - Add 678
 - AddFont 678
 - AddFontEx 678
 - AssignTo 678
 - Create 678
 - FindStyleWithCharset 679
 - FindStyleWithColor 679
 - FindStyleWithFont 679
 - FindStyleWithFontName 680
 - FindStyleWithFontSize 680
 - FindStyleWithFontStyle 681
 - FindSuchStyle 681
 - FindSuchStyleEx 681
 - InvalidItem 676
 - Items 677
 - PixelsPerInch 677
- Title 419
 - TRVDocParameters 419
- TitlePage 419,769
 - TRVDocParameters 419
 - TRVPrint 769
- TJumpEvent 382
- TopLevelEditor 481
 - TRichViewEdit 481
- TopMargin 255,420
 - TRichView 255
 - TRVDocParameters 420
- TParaInfo 727
 - Assign 730
 - Create 731
 - DefStyleNo 729
 - IsEqual 731
 - ModifiedProperties 729
 - NextParaNo 730
- TParaInfos 682
 - Add 684
 - AssignTo 684
 - Create 684

- TParaInfos.FindStyleWithAlignment 684
- TParaInfos.FindSuchStyle 685
- TParaInfos.FindSuchStyleEx 685
- TParaInfos.InvalidItem 683
- TParaInfos.Items 683
- Tracking 818
 - TRVScroller 818
- TransparentBackground 827
 - TCustomRVPrint 827
- TRichView 210, 677
 - adding items 102
 - Component Editor 218
 - live spelling check 132
 - Loading UTF-8 files 460
 - modifying items 112
 - New in v1.2 82
 - New in v1.3 81
 - New in v1.4 80
 - New in v1.5 77
 - New in v1.6 75
 - New in v1.7 72
 - New in v1.8 70
 - New in v1.9 67
 - New in v10 63
 - New in v11 60
 - New in v12 59
 - New in v13 57
 - New in v14 53
 - New in v15 51
 - New in v16 48
 - New in v17 46
 - New in v18 43
 - New in v19 42
 - New in v20 40
 - New in version 21 38
 - New in version 22 36
 - New in version 23 34
 - obtaining items 111
 - selecting parts of document 107
 - Version history 34
- TRichView Clipboard functions overview 108
- TRichView constants and variables (List) 1035
 - CFRV_HTML 1037
 - CFRV_RTF 1037
 - CFRV_RVF 1037
 - CFRV_URL 1037
 - color constants 1038
 - DefaultRVFPixelsPerInch 1042
 - HTMLFormatName 1037
 - pen style constants 1042
 - RichViewAllowCopyTableCells 1042
 - RichViewAlternativePicPrint 1043
 - RichViewAutoAssignNormalStyleTemplate 1043
 - RichViewCompareStyleNames 1044
 - RichViewCSSUseSystemColors 1044
 - RichViewCtrlUpDownScroll 1044
 - RichViewDoNotCheckRVFStyleRefs 1045
 - RichViewDoNotMergeNumbering 1045
 - RichViewEditDefaultProportionalResize 1045
 - RichViewEditEnterAllowsEmptyMarkedLines 1046
 - RichViewJustifyBeforeLineBreak 1046
 - RichViewMaxPictureCount 1046
 - RichViewMaxThumbnailCount 1047
 - RichViewPixelsPerInch 1047
 - RichViewResetStandardFlag 1047
 - RichViewSaveDivInHTMLEx 1040
 - RichViewSaveHyperlinksInDocXAsFields 1048
 - RichViewSavePageBreaksInText 1040
 - RichViewSavePinHTML 1040
 - RichViewShowGhostSpaces 1048
 - RichViewStringFormatMethod 1048
 - RichViewTableAutoAddRow 1049
 - RichViewTableDefaultRTFAutofit 1049
 - RichViewUnicodeInput 1050
 - RTFormatName 1037
 - RVAIwaysShowPlaceholders 1051
 - rveipcAlignment 1051, 1056
 - rveipcBreakColor 1051
 - rveipcBreakStyle 1051
 - rveipcBreakWidth 1051
 - rveipcCursor 1051, 1056
 - rveipcHotImageIndex 1051
 - rveipcImageIndex 1051
 - rveipcMinWidth 1051, 1056
 - rveipcPercentWidth 1051
 - rveipcProtectTextStyleNo 1051, 1056
 - rveipcRemoveInternalLeading 1051, 1056
 - rveipcSeqNumberType 1051, 1056
 - rveipcSeqReset 1051, 1056
 - rveipcSeqStartFrom 1051, 1056
 - RVFormatName 1037
 - RVGetSystemColor 1057
 - RVIscustomURL 1058
 - RVParagraphMarks 1058
 - rsvBreak 1059
 - rsvBullet 1059
 - rsvComponent 1059
 - rsvDefStyle 1059
 - rsvEndnote 1059
 - rsvFootnote 1059
 - rsvHeading 1059

- TRichView constants and variables
 - rvsHotPicture 1059
 - rvsHotspot 1059
 - rvsJump1 1059
 - rvsJump2 1059
 - rvsKeyword 1059
 - rvsLabel 1059
 - rvsListMarker 1059
 - rvsNormal 1059
 - rvsNoteReference 1059
 - rvsPicture 1059
 - rvsSequence 1059
 - rvsSubheading 1059
 - rvsTab 1059
 - rvsTable 1059
 - RVVisibleSpecialCharacters 1061
 - URLFormatName 1037
- TRichView hypertext 92
- TRichView item types 85, 158, 844
- TRichView items' "checkpoints" overview 87
- TRichView items' "tags" overview 91
- TRichView methods for saving and loading 122
- TRichView overview 32, 84
- TRichView paragraphs 95
- TRichView procedures and functions
 - DrawButton 981
 - DrawControl 981
 - DrawEdit 981
 - DrawMemo 981
 - DrawPanel 981
 - GetRVESearchOptions 986
 - GetRVSearchOptions 987
 - RV_GetPrinterDC 988
 - RV_TestFileUnicode 992
 - RV_TestStreamUnicode 992
 - RV_TestStringUnicode 992
 - RVGetFirstEndnote 989
 - RVGetFirstFootnote 989
 - RVGetFirstSidenote 989
 - RVGetNextEndnote 989
 - RVGetNextFootnote 989
 - RVGetNextSidenote 989
 - RVGetNoteTextStyleNo 990
 - RVIsEmail 991
 - RVIsURL 991
 - RVMathEquationManager 991
 - RVOpenURL 991
 - RVU_AnsiToUnicode 992
 - RVU_AnsiToUTF8 992
 - RVU_GetRawUnicode 992
 - RVU_RawUnicodeToWideString 992
 - RVU_UnicodeToAnsi 992
- TRichView scrolling 105
- TRichView.Add 270
- TRichView.AddBreak 262
- TRichView.AddBreakEx 262
- TRichView.AddBreakExTag 262
- TRichView.AddBreakTag 262
- TRichView.AddBullet 263
- TRichView.AddBulletEx 263
- TRichView.AddBulletExTag 263
- TRichView.AddCheckpoint 264
- TRichView.AddCheckpointTag 264
- TRichView.AddControl 265
- TRichView.AddControlEx 265
- TRichView.AddControlExTag 265
- TRichView.AddFmt 266
- TRichView.AddHotPicture 267
- TRichView.AddHotPictureTag 267
- TRichView.AddHotspot 268
- TRichView.AddHotspotEx 268
- TRichView.AddHotspotExTag 268
- TRichView.AddItem 269
- TRichView.AddNamedCheckpoint 264
- TRichView.AddNamedCheckpointEx 264
- TRichView.AddNamedCheckpointExTag 264
- TRichView.AddNL 270
- TRichView.AddNLATag 270
- TRichView.AddNLTag 270
- TRichView.AddNLWTag 270
- TRichView.AddPicture 272
- TRichView.AddPictureEx 272
- TRichView.AddPictureExTag 272
- TRichView.AddTab 273
- TRichView.AddTag 270
- TRichView.AddTextNL 274
- TRichView.AddTextNLA 274
- TRichView.AddTextNLW 274
- TRichView.AllowSelection 222
- TRichView.AnimationMode 222
- TRichView.AppendFrom 275
- TRichView.AppendRVFFromStream 276
- TRichView.BackgroundBitmap 222
- TRichView.BackgroundColor 223
- TRichView.BeginOleDrag 278
- TRichView.BiDiMode 224
- TRichView.BottomMargin 225
- TRichView.Clear 279
- TRichView.ClearLeft 225
- TRichView.ClearLiveSpellingResults 279

TRichView.ClearRight 226
TRichView.ClearSoftPageBreaks 279
TRichView.ClientToDocument 280
TRichView.Color 226
TRichView.ConvertDocToDifferentUnits 280
TRichView.ConvertDocToEMU 280
TRichView.ConvertDocToPixels 280
TRichView.ConvertDocToTwips 280
TRichView.Copy 281
TRichView.CopyDef 281
TRichView.CopyImage 281
TRichView.CopyRTF 282
TRichView.CopyRVF 282
TRichView.CopyText 283
TRichView.CopyTextA 283
TRichView.CopyTextW 283
TRichView.CPEventKind 227
TRichView.Create 283
TRichView.Cursor 227
TRichView.DarkMode 228
TRichView.DeleteItems 283
TRichView.DeleteMarkedStyles 284
TRichView.DeleteParas 285
TRichView.DeleteSection 285
TRichView.DeleteUnusedStyles 286
TRichView.Delimiters 228
TRichView.Deselect 286
TRichView.Destroy 286
TRichView.DocObjects 229
TRichView.DocParameters 230
TRichView.DocProperties 231
TRichView.Document 232
TRichView.DocumentHeight 233
TRichView.DocumentPixelsPerInch 233
TRichView.DocumentToClient 280
TRichView.DoInPaletteMode 234
TRichView.FindCheckpointByName 287
TRichView.FindCheckpointByTag 287
TRichView.FindControlItemNo 287
TRichView.FirstItemVisible 236
TRichView.FirstJumpNo 236
TRichView.Format 288
TRichView.FormatTail 288
TRichView.GetBreakInfo 289
TRichView.GetBulletInfo 289
TRichView.GetCheckpointByNo 291
TRichView.GetCheckpointInfo 291
TRichView.GetCheckpointItemNo 291
TRichView.GetCheckpointNo 292
TRichView.GetCheckpointXY 292
TRichView.GetCheckpointY 292
TRichView.GetCheckpointYEx 293
TRichView.GetControlInfo 293
TRichView.GetFirstCheckpoint 294
TRichView.GetFocusedItem 295
TRichView.GetHotspotInfo 295
TRichView.GetItem 296
TRichView.GetItemCheckpoint 298
TRichView.GetItemClientCoords 298
TRichView.GetItemCoords 298
TRichView.GetItemCoordsEx 299
TRichView.GetItemExtraIntProperty 300
TRichView.GetItemExtraIntPropertyEx 300
TRichView.GetItemNo 301
TRichView.GetItemPara 301
TRichView.GetItemStyle 302
TRichView.GetItemTag 302
TRichView.GetItemText 303
TRichView.GetItemTextA 303
TRichView.GetItemTextW 303
TRichView.GetItemVAlign 303
TRichView.GetJumpPointItemNo 304
TRichView.GetJumpPointLocation 304
TRichView.GetJumpPointY 305
TRichView.GetLastCheckpoint 305
TRichView.GetLineNo 306
TRichView.GetListMarkerInfo 307
TRichView.GetNextCheckpoint 307
TRichView.GetOffsAfterItem 308
TRichView.GetOffsBeforeItem 309
TRichView.GetPictureInfo 310
TRichView.GetPrevCheckpoint 311
TRichView.GetRealDocumentPixelsPerInch 311
TRichView.GetSelectedImage 312
TRichView.GetSelectionBounds 312
TRichView.GetSelText 313
TRichView.GetSelTextA 313
TRichView.GetSelTextW 313
TRichView.GetTextInfo 313
TRichView.GetWordAt 314
TRichView.GetWordAtA 314
TRichView.GetWordAtW 314
TRichView.HTMLReadProperties 237
TRichView.HTMLSaveProperties 237
TRichView.InsertRVFFromStream 316
TRichView.IsFromNewLine 317
TRichView.IsParaStart 318
TRichView.ItemCount 237

- TRichView.LastItemVisible 238
- TRichView.LeftMargin 238
- TRichView.LineCount 238
- TRichView.LiveSpellingValidateWord 318
- TRichView.LoadDocX 318
- TRichView.LoadDocXFromStream 319
- TRichView.LoadFromFile 320
- TRichView.LoadFromFileEx 320
- TRichView.LoadFromStream 321
- TRichView.LoadFromStreamEx 321
- TRichView.LoadHTML 322
- TRichView.LoadHTMLFromStream 324
- TRichView.LoadMarkdown 325
- TRichView.LoadMarkdownFromStream 325
- TRichView.LoadRTF 326
- TRichView.LoadRTFFromStream 327
- TRichView.LoadRVF 328
- TRichView.LoadRVFFromStream 329
- TRichView.LoadText 330
- TRichView.LoadTextFromStream 331
- TRichView.LoadTextFromStreamW 331
- TRichView.LoadTextW 330
- TRichView.MarkdownProperties 239
- TRichView.MarkStylesInUse 331
- TRichView.MaxLength 239
- TRichView.MaxTextWidth 239
- TRichView.MinTextWidth 240
- TRichView.NoteText 240
- TRichView.OnAddStyle 370
- TRichView.OnAfterDrawImage 370
- TRichView.OnAssignImageFileName 371
- TRichView.OnCheckpointVisible 372
- TRichView.OnCMHintShow 372
- TRichView.OnControlAction 373
- TRichView.OnCopy 374
- TRichView.OnGetItemCursor 374
- TRichView.OnGetSpellingSuggestions 375
- TRichView.OnHTMLSaveImage 375
- TRichView.OnImportFile 378
- TRichView.OnImportPicture 378
- TRichView.OnItemAction 380
- TRichView.OnItemHint 381
- TRichView.OnJump 382
- TRichView.OnLoadCustomFormat 383
- TRichView.OnLoadDocument 384
- TRichView.OnNewDocument 384
- TRichView.OnPaint 384
- TRichView.OnProgress 385
- TRichView.OnRVDbClick 391
- TRichView.OnRVFControlNeeded 392
- TRichView.OnRVFImageListNeeded 393
- TRichView.OnRVFPictureNeeded 393
- TRichView.OnRVMouseDown 394
- TRichView.OnRVMouseMove 395
- TRichView.OnRVMouseUp 396
- TRichView.OnRVRightClick 397
- TRichView.OnSaveComponentToFile 397
- TRichView.OnSaveDocXExtra 399
- TRichView.OnSaveHTMLExtra 401
- TRichView.OnSaveImage2 402
- TRichView.OnSaveItemToFile 404
- TRichView.OnSaveParaToHTML 406
- TRichView.OnSaveRTFExtra 406
- TRichView.OnSelect 408
- TRichView.OnSpellingCheck 409
- TRichView.OnStyleTemplatesChange 410
- TRichView.OnTextFound 410
- TRichView.OnWriteHyperlink 411
- TRichView.OnWriteObjectProperties 413
- TRichView.Options 240
- TRichView.PageBreaksBeforeItems 244
- TRichView.Reformat 332
- TRichView.RefreshListMarkers 332
- TRichView.RefreshSequences 332
- TRichView.RemoveCheckpoint 332
- TRichView.ResetAnimation 333
- TRichView.RightMargin 244
- TRichView.RTFOptions 245
- TRichView.RTFReadProperties 246
- TRichView.RVData 247
- TRichView.RVFOptions 247
- TRichView.RVFParaStylesReadMode 251
- TRichView.RVFTextStylesReadMode 251
- TRichView.RVFWarnings 251
- TRichView.SaveDocX 333
- TRichView.SaveDocXToStream 334
- TRichView.SaveHTML 335, 336
- TRichView.SaveHTMLEx 337
- TRichView.SaveHTMLToStream 338, 340
- TRichView.SaveHTMLToStreamEx 340
- TRichView.SaveMarkdown 340
- TRichView.SaveMarkdownToStream 341
- TRichView.SavePicture 342
- TRichView.SaveRTF 343
- TRichView.SaveRTFToStream 344
- TRichView.SaveRVF 344
- TRichView.SaveRVFToStream 344
- TRichView.SaveText 345

-
- TRichView.SaveTextToStream 345
 - TRichView.SaveTextToStreamW 345
 - TRichView.SaveTextW 345
 - TRichView.SearchText 346
 - TRichView.SearchTextA 346
 - TRichView.SearchTextW 346
 - TRichView.SelectAll 348
 - TRichView.SelectControl 348
 - TRichView.SelectionExists 349
 - TRichView.SelectionHandlesVisible 253
 - TRichView.SelectWordAt 349
 - TRichView.SetAddParagraphMode 350
 - TRichView.SetBreakInfo 351
 - TRichView.SetBulletInfo 352
 - TRichView.SetCheckpointInfo 353
 - TRichView.SetControlInfo 354
 - TRichView.SetFooter 355
 - TRichView.SetHeader 356
 - TRichView.SetHotspotInfo 357
 - TRichView.SetItemExtraIntProperty 358
 - TRichView.SetItemExtraIntPropertyEx 358
 - TRichView.SetItemExtraStrProperty 359
 - TRichView.SetItemExtraStrPropertyEx 359
 - TRichView.SetItemTag 360
 - TRichView.SetItemText 360
 - TRichView.SetItemTextA 360
 - TRichView.SetItemTextW 360
 - TRichView.SetItemVAlign 361
 - TRichView.SetListMarkerInfo 362
 - TRichView.SetPictureInfo 363
 - TRichView.SetSelectionBounds 365
 - TRichView.SingleClick 253
 - TRichView.StartAnimation 366
 - TRichView.StartLiveSpelling 367
 - TRichView.StopAnimation 367
 - TRichView.StoreSearchResult 367
 - TRichView.Style 253
 - TRichView.StyleTemplateInsertMode 254
 - TRichView.TabNavigation 255
 - TRichView.TopMargin 255
 - TRichView.UpdatePalettInfo 368
 - TRichView.UseFMXThemes 256
 - TRichView.UseStyleTemplates 256
 - TRichView.UseVCLThemes 257
 - TRichView.VAlign 257
 - TRichView.VSmallStep 257
 - TRichView.WordWrap 258
 - TRichView.ZoomPercent 258
 - TRichViewEdit 461, 533
 - Inserting items at position of caret 114
 - InsertTextFromFile 533
 - InsertTextFromFileW 533
 - Moving caret to the beginning of paragraph 592
 - OnDropFiles example 593
 - TRichViewEdit Undo 115
 - TRichViewEdit.AcceptDragDropFormats 470
 - TRichViewEdit.AcceptPasteFormats 470
 - TRichViewEdit.AddSpellingMenuItems 488
 - TRichViewEdit.AdjustControlPlacement 489
 - TRichViewEdit.AdjustControlPlacement2 489
 - TRichViewEdit.ApplyListStyle 490
 - TRichViewEdit.ApplyParaStyle 490
 - TRichViewEdit.ApplyParaStyleConversion 491
 - TRichViewEdit.ApplyParaStyleTemplate 491
 - TRichViewEdit.ApplyStyleConversion 492
 - TRichViewEdit.ApplyStyleTemplate 493
 - TRichViewEdit.ApplyTextStyle 494
 - TRichViewEdit.ApplyTextStyleTemplate 494
 - TRichViewEdit.BeginItemModify 495
 - TRichViewEdit.BeginUndoCustomGroup 496
 - TRichViewEdit.BeginUndoCustomGroup2 496
 - TRichViewEdit.BeginUndoGroup 497
 - TRichViewEdit.BeginUndoGroup2 497
 - TRichViewEdit.BeginUpdate 498
 - TRichViewEdit.CanChange 498
 - TRichViewEdit.CanPaste 498
 - TRichViewEdit.CanPasteHTML 499
 - TRichViewEdit.CanPasteRTF 499
 - TRichViewEdit.CanPasteRVF 499
 - TRichViewEdit.Change 499
 - TRichViewEdit.ChangeListLevels 500
 - TRichViewEdit.ChangeStyleTemplates 500
 - TRichViewEdit.CheckSpelling 472
 - TRichViewEdit.Clear 501
 - TRichViewEdit.ClearTextFlow 501
 - TRichViewEdit.ClearUndo 501
 - TRichViewEdit.ConvertToHotPicture 501
 - TRichViewEdit.ConvertToPicture 502
 - TRichViewEdit.Create 502
 - TRichViewEdit.CurItemNo 472
 - TRichViewEdit.CurItemStyle 473
 - TRichViewEdit.CurParaStyleNo 473
 - TRichViewEdit.CurTextStyleNo 474
 - TRichViewEdit.CustomCaretInterval 474
 - TRichViewEdit.CutDef 502
 - TRichViewEdit.DefaultPictureVAlign 475
 - TRichViewEdit.DeleteSelection 503
 - TRichViewEdit.Destroy 503

- TRichViewEdit.EditorOptions 475
TRichViewEdit.EndItemModify 503
TRichViewEdit.EndUndoCustomGroup2 496
TRichViewEdit.EndUndoGroup2 497
TRichViewEdit.EndUpdate 503
TRichViewEdit.ForceFieldHighlight 478
TRichViewEdit.GetCheckpointAtCaret 503
TRichViewEdit.GetCurrentBreakInfo 504
TRichViewEdit.GetCurrentBulletInfo 505
TRichViewEdit.GetCurrentCheckpoint 506
TRichViewEdit.GetCurrentControlInfo 506
TRichViewEdit.GetCurrentHotspotInfo 507
TRichViewEdit.GetCurrentItem 508
TRichViewEdit.GetCurrentItemEx 509
TRichViewEdit.GetCurrentItemExtraIntProperty 510
TRichViewEdit.GetCurrentItemExtraIntPropertyEx 510
TRichViewEdit.GetCurrentItemExtraStrProperty 510
TRichViewEdit.GetCurrentItemExtraStrPropertyEx 510
TRichViewEdit.GetCurrentItemText 511
TRichViewEdit.GetCurrentItemTextA 511
TRichViewEdit.GetCurrentItemTextW 511
TRichViewEdit.GetCurrentItemVAlign 512
TRichViewEdit.GetCurrentLineCol 512
TRichViewEdit.GetCurrentMisspelling 513
TRichViewEdit.GetCurrentPictureInfo 513
TRichViewEdit.GetCurrentTag 514
TRichViewEdit.GetCurrentTextInfo 514
TRichViewEdit.InsertBreak 515
TRichViewEdit.InsertBullet 515
TRichViewEdit.InsertCheckpoint 516
TRichViewEdit.InsertControl 517
TRichViewEdit.InsertDocXFromFileEd 518
TRichViewEdit.InsertDocXFromStreamEd 518
TRichViewEdit.InsertHotPicture 519
TRichViewEdit.InsertHotspot 520
TRichViewEdit.InsertHTMLFromFileEd 521
TRichViewEdit.InsertHTMLFromStreamEd 521
TRichViewEdit.InsertItem 522
TRichViewEdit.InsertMarkdownFromFileEd 523
TRichViewEdit.InsertMarkdownFromStreamEd 524
TRichViewEdit.InsertOEMTextFromFile 525
TRichViewEdit.InsertPageBreak 525
TRichViewEdit.InsertPicture 526
TRichViewEdit.InsertRTFFFromFileEd 527
TRichViewEdit.InsertRTFFFromStreamEd 528
TRichViewEdit.InsertRVFFFromFileEd 528
TRichViewEdit.InsertRVFFFromStreamEd 529
TRichViewEdit.InsertStringATag 530
TRichViewEdit.InsertStringTag 530
TRichViewEdit.InsertStringWTag 530
TRichViewEdit.InsertTab 531
TRichViewEdit.InsertText 532
TRichViewEdit.InsertTextA 532
TRichViewEdit.InsertTextW 532
TRichViewEdit.KeyboardType 478
TRichViewEdit.LiveSpellingMode 479
TRichViewEdit.Modified 479
TRichViewEdit.MoveCaret 534
TRichViewEdit.OffsetInCurlItem 479
TRichViewEdit.OnAfterOleDrop 570
TRichViewEdit.OnBeforeOleDrop 570
TRichViewEdit.OnCaretGetOut 571
TRichViewEdit.OnCaretMove 572
TRichViewEdit.OnChange 572
TRichViewEdit.OnChanging 572
TRichViewEdit.OnCheckStickingItems 573
TRichViewEdit.OnCurParaStyleChanged 573
TRichViewEdit.OnCurTextStyleChanged 574
TRichViewEdit.OnDrawCurrentLine 574
TRichViewEdit.OnDrawCustomCaret 575
TRichViewEdit.OnDropFile 576
TRichViewEdit.OnDropFiles 576
TRichViewEdit.OnItemResize 577
TRichViewEdit.OnItemTextEdit 578
TRichViewEdit.OnMeasureCustomCaret 578
TRichViewEdit.OnOleDragEnter 579
TRichViewEdit.OnOleDragLeave 579
TRichViewEdit.OnOleDragOver 580
TRichViewEdit.OnOleDrop 581
TRichViewEdit.OnParaStyleConversion 582
TRichViewEdit.OnPaste 584
TRichViewEdit.OnSaveCustomFormat 585
TRichViewEdit.OnSmartPopupClick 585
TRichViewEdit.OnStyleConversion 585
TRichViewEdit.Paste 534
TRichViewEdit.PasteBitmap 535
TRichViewEdit.PasteGraphicFile 535
TRichViewEdit.PasteHTML 536
TRichViewEdit.PasteMetafile 537
TRichViewEdit.PasteRTF 537
TRichViewEdit.PasteRVF 538
TRichViewEdit.PasteText 538
TRichViewEdit.PasteTextA 538
TRichViewEdit.PasteTextW 538
TRichViewEdit.PasteURL 539
TRichViewEdit.ReadOnly 480
TRichViewEdit.Redo 540
TRichViewEdit.RedoAction 540

TRichViewEdit.RedoName	540	TRichViewEdit.SmartPopupVisible	481
TRichViewEdit.RemoveCheckpointAtCaret	541	TRichViewEdit.TopLevelEditor	481
TRichViewEdit.RemoveCheckpointEd	541	TRichViewEdit.Undo	567
TRichViewEdit.RemoveCurrentCheckpoint	541	TRichViewEdit.UndoAction	567
TRichViewEdit.RemoveCurrentPageBreak	542	TRichViewEdit.UndoLimit	481
TRichViewEdit.RemoveLists	542	TRichViewEdit.UndoName	568
TRichViewEdit.ResizeControl	542	TRichViewRVData	957
TRichViewEdit.ResizeCurrentControl	543	TRichViewUnicodeInput	1050
TRichViewEdit.SearchText	543	TRV_AfterImportGraphicsProc	1068
TRichViewEdit.SearchTextA	543	TRV_CreateGraphicsFunction	1068
TRichViewEdit.SearchTextW	543	TRVAddStyleEvent	370
TRichViewEdit.SelectCurrentLine	545	TRVAfterApplyStyleEvent	664
TRichViewEdit.SelectCurrentWord	545	TRVAfterDrawImageEvent	370
TRichViewEdit.SetBackgroundImageEd	546	TRVAfterPrintImageEvent	829
TRichViewEdit.SetBreakInfoEd	547	TRVAlignment	719
TRichViewEdit.SetBulletInfoEd	548	TRVAnimationMode	222
TRichViewEdit.SetCheckpointInfoEd	549	TRVAnsiString	993
TRichViewEdit.SetControlInfoEd	550	TRVApplyStyleColorEvent	666
TRichViewEdit.SetCurrentBreakInfo	551	TRVApplyStyleEvent	665
TRichViewEdit.SetCurrentBulletInfo	552	TRVAssignImageFileNameEvent	371
TRichViewEdit.SetCurrentCheckpointInfo	553	TRVBackgroundRect	741
TRichViewEdit.SetCurrentControlInfo	554	TRVBackgroundRect.BorderOffsets	742
TRichViewEdit.SetCurrentHotspotInfo	555	TRVBackgroundRect.Color	742
TRichViewEdit.SetCurrentItemExtraIntProperty	556	TRVBackgroundRect.Opacity	742
TRichViewEdit.SetCurrentItemExtraIntPropertyEx	556	TRVBiDiMode	993
TRichViewEdit.SetCurrentItemExtraStrProperty	557	TRVBooleanRect	961
TRichViewEdit.SetCurrentItemExtraStrPropertyEx	557	TRVBorder	742
TRichViewEdit.SetCurrentItemText	557	TRVBorder.BorderOffsets	743
TRichViewEdit.SetCurrentItemTextA	557	TRVBorder.Color	744
TRichViewEdit.SetCurrentItemTextW	557	TRVBorder.InternalWidth	744
TRichViewEdit.SetCurrentItemVAlign	558	TRVBorder.Style	744
TRichViewEdit.SetCurrentPictureInfo	559	TRVBorder.VisibleBorders	744
TRichViewEdit.SetCurrentTag	560	TRVBorder.Width	744
TRichViewEdit.SetFloatPropertyEd	545	TRVBorderStyle	994
TRichViewEdit.SetHotspotInfoEd	560	TRVBoxBackgroundRect	938
TRichViewEdit.SetIntPropertyEd	545	TRVBoxBorder	938
TRichViewEdit.SetItemExtraIntPropertyEd	562	TRVBoxHeightType	938
TRichViewEdit.SetItemExtraIntPropertyExEd	562	TRVBoxPosition	940
TRichViewEdit.SetItemExtraStrPropertyEd	562	TRVBoxPosition.HorizontalAlignment	942
TRichViewEdit.SetItemExtraStrPropertyExEd	562	TRVBoxPosition.HorizontalAnchor	941
TRichViewEdit.SetItemTagEd	563	TRVBoxPosition.HorizontalOffset	942
TRichViewEdit.SetItemTextEd	564	TRVBoxPosition.HorizontalPositionKind	942
TRichViewEdit.SetItemTextEdA	564	TRVBoxPosition.PositionInText	944
TRichViewEdit.SetItemTextEdW	564	TRVBoxPosition.RelativeToCell	945
TRichViewEdit.SetItemVAlignEd	565	TRVBoxPosition.VerticalAlignment	946
TRichViewEdit.SetPictureInfoEd	565	TRVBoxPosition.VerticalAnchor	945
TRichViewEdit.SetStrPropertyEd	545	TRVBoxPosition.VerticalOffset	946
TRichViewEdit.SetUndoGroupMode	567	TRVBoxPosition.VerticalPositionKind	946
TRichViewEdit.SmartPopupProperties	480	TRVBoxProperties	937

- TRVBoxProperties.Background 938
 TRVBoxProperties.Border 938
 TRVBoxProperties.Height 938
 TRVBoxProperties.HeightType 938
 TRVBoxProperties.VAlign 939
 TRVBoxProperties.Width 939
 TRVBoxProperties.WidthType 939
 TRVBoxWidthType 939
 TRVBreakItemInfo 844
 TRVBreakStyle 995
 TRVBulletItemInfo 844
 TRVCanvas 961
 TRVCaretMovement 534
 TRVCaretWidthMode 1040
 TRVCellEditingEvent 892
 TRVCellVAlign 996
 TRVCheckPointVisibleEvent 372
 TRVClickMode 836
 TRVCMHintShowEvent 372
 TRVCodePage 992, 996
 TRVColor 996
 TRVColorMode 822, 997
 TRVControlAction 373
 TRVControlActionEvent 373
 TRVControlData 957
 TRVControlItemInfo 844
 TRVControlsPainter 962
 TRVCoord 998
 TRVCoordPoint 998
 TRVCoordRect 998
 TRVCoordSize 998
 TRVCSSSavingType 431
 TRVCustomFormatEvent 383, 585
 TRVCustomMathEquationManager 968
 SaveMathMLInHTML 969
 SaveOMMLInDocX 969
 TRVCustomMathEquationManager.SaveMathMLInHTML 969
 TRVCustomMathEquationManager.SaveOMMLInDocX 969
 TRVDataSourceLink 811
 TRVDataSourceLink.DataSource 812
 TRVDataSourceLink.Editor 812
 TRVDataSourceLink.Execute 813
 TRVDBFieldFormat 998
 TRVDbClickEvent 391
 TRVDefaultLoadProperties 963
 TRVDefaultLoadProperties.DefaultBulletFontName 963
 TRVDefaultLoadProperties.DefaultFontName 964
 TRVDefaultLoadProperties.DefaultFontSizeDouble 964
 TRVDefaultLoadProperties.DefaultFontSizesDouble 964
 TRVDefaultLoadProperties.DefaultMonoSpacedFontName 965
 TRVDefaultLoadProperties.DefaultProperties 965
 TRVDefaultLoadProperties.DefaultUnicodeSymbolFontName 965
 TRVDefaultLoadProperties.GenericFontNames 965
 TRVDefaultLoadProperties.ListMarkerIndentPix 968
 TRVDeleteUnusedStylesData 969
 TRVDisplayOption 999
 TRVDisplayOptions 999
 TRVDocObject 229
 TRVDocObjectCollection 229
 TRVDocParameters 230, 415
 TRVDocParameters.Author 416
 TRVDocParameters.BottomMargin 416
 TRVDocParameters.Comments 417
 TRVDocParameters.ConvertToUnit 421
 TRVDocParameters.FacingPages 417
 TRVDocParameters.FooterY 417
 TRVDocParameters.HeaderY 418
 TRVDocParameters.LeftMargin 418
 TRVDocParameters.MirrorMargins 418
 TRVDocParameters.Orientation 418
 TRVDocParameters.PageHeight 419
 TRVDocParameters.PageWidth 419
 TRVDocParameters.Reset 421
 TRVDocParameters.ResetLayout 421
 TRVDocParameters.RightMargin 419
 TRVDocParameters.Title 419
 TRVDocParameters.TitlePage 419
 TRVDocParameters.TopMargin 420
 TRVDocParameters.Units 420
 TRVDocParameters.UnitsPerInch 421
 TRVDocParameters.ZoomMode 420
 TRVDocParameters.ZoomPercent 421
 TRVDocRotation 999
 TRVDocumentProperty 422
 TRVDocumentProperty.AllowMarkdown 422
 TRVDocumentProperty.AutoDeleteUnusedStyles 423
 TRVDocumentProperty.CodePage 423
 TRVDocumentProperty.FieldFormat 424
 TRVDocXErrorCode 454
 TRVDocXSaveArea 399
 TRVDragDropFormat 470
 TRVDragDropFormats 470
 TRVDrawCheckpointEvent 667, 830
 TRVDrawCurrentLineEvent 574

-
- TRVDrawCustomCaretEvent 575
 - TRVDrawHyperLinkEvent 830
 - TRVDrawPageBreakEvent 668
 - TRVDrawParaRectEvent 670
 - TRVDrawStyleTextEvent 671
 - TRVDrawTextBackEvent 673
 - TRVDropFileAction 576
 - TRVDropFileEvent 576
 - TRVDropFilesEvent 576
 - TRVEditRVData 957
 - TRVEndnoteItemInfo 844, 928
 - TRVEndnoteItemInfo.Create 930
 - TRVEndnoteItemInfo.CreateEx 930
 - TRVEnumScope 999
 - TRVSearchOption 999
 - TRVSearchOptions 999
 - TRVExtraItemProperty 1000
 - TRVExtraItemStrProperty 1005
 - TRVControlNeededEvent 392
 - TRVFieldContext 386, 390
 - TRVFieldResultData 386, 390
 - TRVImageListNeededEvent 393
 - TRVFixMarginsMode 766
 - TRVFloatHorizontalAlignment 942
 - TRVFloatPosition 1006
 - TRVFloatPositionInText 944
 - TRVFloatPositionKind 1006
 - TRVFloatProperty 1007
 - TRVFloatSize 1007
 - TRVFloatVerticalAlignment 946
 - TRVFMXCanvas 961
 - TRVFontCharset 1008
 - TRVFontInfoClass 661
 - TRVFontInfoProperties 1008
 - TRVFontInfoProperty 1008
 - TRVFontSize 1009
 - TRVFontSizePrecision 426
 - TRVFontStyle 708
 - TRVFontStyles 708
 - TRVFootnoteItemInfo 844, 930, 933
 - TRVFootnoteItemInfo.Create 932
 - TRVFootnoteItemInfo.CreateEx 933
 - TRVFootOrEndnoteItemInfo 925
 - TRVFOption 247
 - TRVFOptions 247
 - TRVFormattingDefaultItem 965
 - TRVFormattingType 965
 - TRVFPictureNeededEvent 393
 - TRVFRReaderStyleMode 1009
 - TRVFullLineItemInfo 844
 - TRVFWarning 251
 - TRVFWarnings 251
 - TRVGetItemCursorEvent 374
 - TRVGetOutDirection 571
 - TRVGetPageNumberEvent 811
 - TRVGetSpellingSuggestionsEvent 375
 - TRVGetSystemColorFunction 1057
 - TRVGraphic 970
 - TRVGraphicClass 971, 972
 - TRVGraphicFM 970
 - TRVGraphicHandler 972
 - TRVGraphicItemInfo 844
 - TRVGraphicType 1010
 - TRVHType 1011
 - TRVHorizontalAnchor 941
 - TRVHorizontalCaretPosition 1040
 - TRVHotGraphicItemInfo 844
 - TRVHotspotItemInfo 844
 - TRVHoverEffect 703
 - TRVHoverEffects 703
 - TRVHTMLCSSSaveOption 430
 - TRVHTMLCSSSaveOptions 430
 - TRVHTMLEncoding 1011
 - TRVHTMLLength 1012
 - TRVHTMListMarkerSavingType 436
 - TRVHTMLReaderProperties 424
 - TRVHTMLReaderProperties.BasePathLinks 425
 - TRVHTMLReaderProperties.EnforceListLevels 426
 - TRVHTMLReaderProperties.FontSizePrecision 426
 - TRVHTMLReaderProperties.IDAsCheckpoints 427
 - TRVHTMLReaderProperties.ReadBackground 427
 - TRVHTMLReaderProperties.ReadDocParameters 427
 - TRVHTMLReaderProperties.SkipHiddenText 428
 - TRVHTMLReaderProperties.XHTMLCheck 428
 - TRVHTMLSaveArea 401
 - TRVHTMLSaveImageEvent 375
 - TRVHTMLSaveImageOption 434
 - TRVHTMLSaveImageOptions 434
 - TRVHTMLSaveProperties 429
 - TRVHTMLSaveProperties.CheckpointsPrefix 430
 - TRVHTMLSaveProperties.CSSOptions 430
 - TRVHTMLSaveProperties.CSSSavingType 431
 - TRVHTMLSaveProperties.Default0Style 431
 - TRVHTMLSaveProperties.Encoding 431
 - TRVHTMLSaveProperties.EncodingStr 432
 - TRVHTMLSaveProperties.ExternalCSSFileName 432
 - TRVHTMLSaveProperties.ExtraStyles 433
 - TRVHTMLSaveProperties.HTMLSavingType 433
-

- TRVHTMLSaveProperties.HTMLVersion 433
 TRVHTMLSaveProperties.ImageOptions 434
 TRVHTMLSaveProperties.ImagesPrefix 436
 TRVHTMLSaveProperties.ListMarkerSavingType 436
 TRVHTMLSaveProperties.SaveHeaderAndFooter 437
 TRVHTMLSaveProperties.UseCheckpointNames 437
 TRVHTMLSavingPart 1012
 TRVHTMLSavingType 433
 TRVHTMLVersion 433
 TRVImportFileEvent 378
 TRVImportPictureEvent 378
 TRVIntegerList 974
 TRVIntProperty 1013
 TRVItemAction 380
 TRVItemActionEvent 380
 TRVItemBackgroundStyle 1013
 TRVItemFormattedData 957
 TRVItemHintEvent 381
 TRVItemResizeEvent 577
 TRVItemTextEditEvent 578
 TRVLabelItemInfo 844, 912
 TRVLabelItemInfo.Alignment 914
 TRVLabelItemInfo.Create 916
 TRVLabelItemInfo.CreateEx 916
 TRVLabelItemInfo.Cursor 914
 TRVLabelItemInfo.MinWidth 914
 TRVLabelItemInfo.ProtectTextStyleNo 914
 TRVLabelItemInfo.RemoveInternalLeading 915
 TRVLabelItemInfo.RemoveSpaceBelow 915
 TRVLabelItemInfo.Text 915
 TRVLabelItemInfo.TextStyleNo 915
 TRVLastLineAlignment 719
 TRVLength 1015
 TRVLineSpacingType 723
 TRVLineSpacingValue 722
 TRVLineWrapMode 644
 TRVListInfo 731
 TRVListInfo.AllNumbered 733
 TRVListInfo.HasNumbering 733
 TRVListInfo.Levels 732
 TRVListInfo.OneLevelPreview 732
 TRVListInfoClass 660
 TRVListInfos 685
 TRVListInfos.Add 687
 TRVListInfos.AssignTo 687
 TRVListInfos.FindStyleWithLevels 687
 TRVListInfos.FindSuchStyle 686
 TRVListInfos.Items 686
 TRVListLevel 750
 TRVListLevel.FirstIndent 751
 TRVListLevel.Font 751
 TRVListLevel.FormatString 752
 TRVListLevel.FormatStringW 753
 TRVListLevel.HasNumbering 759
 TRVListLevel.ImageHeight 754
 TRVListLevel.ImageIndex 753
 TRVListLevel.ImageList 754
 TRVListLevel.ImageWidth 754
 TRVListLevel.LeftIndent 755
 TRVListLevel.ListType 755
 TRVListLevel.MarkerAlignment 757
 TRVListLevel.MarkerIndent 757
 TRVListLevel.Options 758
 TRVListLevel.Picture 759
 TRVListLevel.StartFrom 759
 TRVListLevelCollection 749
 TRVListLevelCollection.Add 749
 TRVListLevelCollection.Items 749
 TRVListLevelOption 758
 TRVListLevelOptions 758
 TRVListType 755
 TRVLiveSpellingMode 479
 TRVLoadFormat 1015
 TRVLongOperation 385
 TRVMarginsPen 837
 TRVMarkdownDefaultItemProperties 438, 444
 TRVMarkdownDefaultItemProperties.BlockQuoteBackColor or 438
 TRVMarkdownDefaultItemProperties.BreakColor 438
 TRVMarkdownDefaultItemProperties.CodeBlockBackColor 439
 TRVMarkdownDefaultItemProperties.ImageVAlign 439
 TRVMarkdownDefaultItemProperties.ListLeftIndentPix 439
 TRVMarkdownDefaultItemProperties.ListMarkerIndentPix 439
 TRVMarkdownDefaultItemProperties.TableCellWidthPix 439
 TRVMarkdownExtension 444
 TRVMarkdownExtensions 444
 TRVMarkdownLoadOption 446
 TRVMarkdownLoadOptions 446
 TRVMarkdownProperties 439
 TRVMarkdownProperties.DefaultItemProperties 444
 TRVMarkdownProperties.Extensions 444
 TRVMarkdownProperties.ImagesPrefix 445
 TRVMarkdownProperties.LineWidth 445
 TRVMarkdownProperties.LoadOptions 446
 TRVMarkdownProperties.NotesSavedAsFootnotes 448

TRVMarkdownProperties.SaveHiddenText	448	TRVOfficeConverterInfo	804
TRVMarkdownProperties.SaveOptions	448	TRVOfficeConverterInfo.Filter	804
TRVMarkdownSaveOption	448	TRVOfficeConverterInfo.Name	804
TRVMarkdownSaveOptions	448	TRVOfficeConverterInfo.Path	804
TRVMarkerAlignment	757	TRVoleDragEnterEvent	579
TRVMarkerItemInfo	844	TRVoleDragOverEvent	580
TRVMathDocObject	974	TRVoleDropEffect	1015
TRVMathItemInfo	917	TRVoleDropEffects	1015
TRVMathItemInfo.Create	920	TRVoleDropEvent	581
TRVMathItemInfo.DisplayInline	918	TRVOnCaretGetOutEvent	571
TRVMathItemInfo.FontName	918	TRVOpacity	1015
TRVMathItemInfo.FontSizeDouble	919	TRVOption	240
TRVMathItemInfo.Text	919	TRVOptions	240
TRVMathItemInfo.TextColor	920	TRVPageBreakType	668
TRVMeasureCustomCaretEvent	578	TRVPageCountItemInfo	954
TRVMouseEvent	394, 396	TRVPageCountItemInfo.Create	955
TRVMouseMoveEvent	395	TRVPageCountItemInfo.CreateEx	955
TRVNonTextItemInfo	844	TRVPageNumberItemInfo	953
TRVNoteData	957	TRVPageNumberItemInfo.Create	953
TRVNoteReferenceltemInfo	844, 949	TRVPageNumberItemInfo.CreateEx	954
TRVNoteReferenceltemInfo.Create	951	TRVPageNumberType	1016
TRVNoteReferenceltemInfo.CreateEx	951	TRVPagePrepaintEvent	781, 782
TRVNoteType	448	TRVPageSet	1016
TRVNoteTypes	448	TRVPaintEvent	370, 384
TRVObjectExportProperties	413	TRVPaletteAction	234
TRVOfficeCnvList	803	TRVPAPen	838
TRVOfficeCnvList.Count	803	TRVParagraphMarks	1058
TRVOfficeCnvList.Items	803	TRVParaInfoClass	661
TRVOfficeConverter	795	TRVParaInfoProperties	1017
TRVOfficeConverter.Create	799	TRVParaInfoProperty	1017
TRVOfficeConverter.Destroy	799	TRVParaOption	723
TRVOfficeConverter.ErrorCode	796	TRVParaOptions	723
TRVOfficeConverter.ExcludeDocXExportConverter	797	TRVPasteEvent	584
TRVOfficeConverter.ExcludeDocXImportConverter	797	TRVPDFMetadata	1018
TRVOfficeConverter.ExcludeHTMLExportConverter	797	TRVPenStyle	1018
TRVOfficeConverter.ExcludeHTMLImportConverter	797	TRVPicture	975
TRVOfficeConverter.ExportConverters	798	TRVPictureFM	975
TRVOfficeConverter.ExportRTF	800	TRVPixel96Length	1018
TRVOfficeConverter.ExportRV	800	TRVPixelLength	1019
TRVOfficeConverter.ExtensionsInFilter	798	TRVPrint	759
TRVOfficeConverter.GetExportFilter	801	TRVPrint.AssignDocParameters	771
TRVOfficeConverter.GetImportFilter	801	TRVPrint.AssignSource	771
TRVOfficeConverter.ImportConverters	798	TRVPrint.BestDPI	765
TRVOfficeConverter.ImportRTF	801	TRVPrint.ClipMargins	765
TRVOfficeConverter.ImportRV	802	TRVPrint.ContinuousPrint	772
TRVOfficeConverter.IsValidImporter	803	TRVPrint.ConvertToUnits	773
TRVOfficeConverter.OnConverting	803	TRVPrint.Create	773
TRVOfficeConverter.PreviewMode	799	TRVPrint.DrawPage	773
TRVOfficeConverter.Stream	799	TRVPrint.DrawPageAt	773

- TRVPrint.DrawPreview 773
- TRVPrint.FacingPages 765
- TRVPrint.FixMarginsMode 766
- TRVPrint.FooterY 766
- TRVPrint.FormatPages 774
- TRVPrint.GetFooterRect 775
- TRVPrint.GetHeaderRect 775
- TRVPrint.HeaderY 767
- TRVPrint.MakePreview 775
- TRVPrint.MakeScaledPreview 776
- TRVPrint.Margins 768
- TRVPrint.MirrorMargins 769
- TRVPrint.OnFormatting 780
- TRVPrint.OnPagePostpaint 781
- TRVPrint.OnPagePrepaint 782
- TRVPrint.OnSendingToPrinter 782
- TRVPrint.Preview100PercentHeight 769
- TRVPrint.Preview100PercentWidth 769
- TRVPrint.Print 776
- TRVPrint.PrintPages 777
- TRVPrint.SavePDF 778
- TRVPrint.SetFooter 778
- TRVPrint.SetHeader 779
- TRVPrint.TitlePage 769
- TRVPrint.Units 770
- TRVPrint.VirtualPrinter 770
- TRVPrintComponentEvent 831
- TRVPrinterDPI 765
- TRVPrintingEvent 780, 782
- TRVPrintingStep 1019
- TRVPrintPreview 625
- TRVPrintPreview.CachePageImage 628
- TRVPrintPreview.RVPrint 628
- TRVProgressEvent 385
- TRVProgressStage 385
- TRVProtectOption 705
- TRVProtectOptions 705
- TRVRawByteString 1019
- TRVReaderStyleMode 1019
- TRVReaderUnicode 459
- TRVReadFieldEvent 386, 390
- TRVReadHyperlink 388
- TRVRect 975
- TRVRectItemInfo 844
- TRVReportHelper 804
- TRVReportHelper.AllowTransparentDrawing 806
- TRVReportHelper.DrawPage 808
- TRVReportHelper.DrawPageAt 808
- TRVReportHelper.Finished 809
- TRVReportHelper.FormatNextPage 809
- TRVReportHelper.GetLastPageHeight 809
- TRVReportHelper.Init 809
- TRVReportHelper.OnGetPageNumber 811
- TRVReportHelper.RichView 807
- TRVReportHelper.SavePDF 810
- TRVReportHelper.TargetPixelsPerInch 806
- TRVReportHelper.UpdatePageNumbers 810
- TRVReportHelperWithHeaderFooters 976
- TRVReportHelperWithHeaderFooters.AssignToRVPrint 977
- TRVReportHelperWithHeaderFooters.Create 977
- TRVReportHelperWithHeaderFooters.GetFooter 978
- TRVReportHelperWithHeaderFooters.GetHeader 978
- TRVReportHelperWithHeaderFooters.MainDoc 977
- TRVReportHelperWithHeaderFooters.Reset 978
- TRVReportHelperWithHeaderFooters.UseStyleTemplates 977
- TRVRightClickEvent 397
- TRVRTFErrorCode 457
- TRVRTFFieldResultAction 386
- TRVRTFHighlight 452
- TRVRTFOption 245
- TRVRTFOptions 245
- TRVRTFReaderProperties 450
- TRVRTFReaderProperties.AutoHideTableGridLines 451
- TRVRTFReaderProperties.BasePathLinks 451
- TRVRTFReaderProperties.CanReuseNumberedLists 452
- TRVRTFReaderProperties.CharsetForUnicode 452
- TRVRTFReaderProperties.ConvertHighlight 452
- TRVRTFReaderProperties.ConvertSymbolFonts 453
- TRVRTFReaderProperties.DocXErrorCode 454
- TRVRTFReaderProperties.ExtractMetafileBitmaps 454
- TRVRTFReaderProperties.IgnoreBookmarks 454
- TRVRTFReaderProperties.IgnoreFields 455
- TRVRTFReaderProperties.IgnoreNotes 455
- TRVRTFReaderProperties.IgnorePictures 455
- TRVRTFReaderProperties.IgnoreSequences 455
- TRVRTFReaderProperties.IgnoreTables 455
- TRVRTFReaderProperties.LineBreaksAsParagraphs 456
- TRVRTFReaderProperties.ParaStyleMode 456
- TRVRTFReaderProperties.ParaStyleNo 457
- TRVRTFReaderProperties.ReadDocParameters 457
- TRVRTFReaderProperties.ReadNoteNumberingType 457
- TRVRTFReaderProperties.RTFErrorCode 457
- TRVRTFReaderProperties.SkipHiddenText 458
- TRVRTFReaderProperties.TextStyleMode 458
- TRVRTFReaderProperties.TextStyleNo 459
- TRVRTFReaderProperties.UnicodeMode 459

-
- TRVRTFReaderProperties.UseCharsetForUnicode 459
 - TRVRTFReaderProperties.UseHypertextStyles 459
 - TRVRTFReaderProperties.UseSingleCellPadding 460
 - TRVRTFSaveArea 406
 - TRVSaveComponentToFileEvent 397
 - TRVSaveCSSOption 662
 - TRVSaveCSSOptions 662
 - TRVSaveDocXExtraEvent 399
 - TRVSaveFormat 1020
 - TRVSaveHTMLExtraEvent 401
 - TRVSaveImageEvent2 402
 - TRVSaveOption 1020
 - TRVSaveOptions 1020
 - TRVSaveParaToHTMLEvent 406
 - TRVSaveRTFExtraEvent 406
 - TRVScreenAndDevice 1024
 - TRVScroller 813
 - TRVScroller.BorderStyle 816
 - TRVScroller.Create 820
 - TRVScroller.HScrollMax 816
 - TRVScroller.HScrollPos 816
 - TRVScroller.HScrollVisible 816
 - TRVScroller.InplaceEditor 817
 - TRVScroller.NoVScroll 817
 - TRVScroller.OnHScrolled 821
 - TRVScroller.OnVScrolled 821
 - TRVScroller.ScrollTo 820
 - TRVScroller.Tracking 818
 - TRVScroller.UseXPThemes 818
 - TRVScroller.VScrollMax 818
 - TRVScroller.VScrollPos 819
 - TRVScroller.VScrollVisible 819
 - TRVScroller.WheelStep 819
 - TRVSearchOption 1024
 - TRVSearchOptions 1024
 - TRVSelection 984
 - TRVSelectionHandleKind 648
 - TRVSelectionMode 649
 - TRVSelectionStyle 649
 - TRVSeqItemInfo 844, 921
 - TRVSeqItemInfo.Create 924
 - TRVSeqItemInfo.CreateEx 925
 - TRVSeqItemInfo.FormatString 923
 - TRVSeqItemInfo.NumberType 923
 - TRVSeqItemInfo.Reset 923
 - TRVSeqItemInfo.SeqName 924
 - TRVSeqItemInfo.StartFrom 924
 - TRVSeqType 1026
 - TRVSidenoteItemInfo.BoxPosition 935
 - TRVSidenoteItemInfo.BoxProperties 935
 - TRVSidenoteItemInfo.Create 936
 - TRVSidenoteItemInfo.CreateEx 936
 - TRVSmartPopupClickEvent 585
 - TRVSmartPopupPosition 591
 - TRVSmartPopupProperties 588
 - TRVSmartPopupProperties.ButtonType 589
 - TRVSmartPopupProperties.Color 589
 - TRVSmartPopupProperties.Hint 590
 - TRVSmartPopupProperties.HoverColor 590
 - TRVSmartPopupProperties.HoverLineColor 590
 - TRVSmartPopupProperties.ImageIndex 590
 - TRVSmartPopupProperties.ImageList 590
 - TRVSmartPopupProperties.LineColor 591
 - TRVSmartPopupProperties.Menu 591
 - TRVSmartPopupProperties.Popup 591
 - TRVSmartPopupProperties.Position 591
 - TRVSmartPopupProperties.SetButtonState 592
 - TRVSmartPopupProperties.ShortCut 591
 - TRVSpecialCharacter 1061
 - TRVSpecialCharacters 1061
 - TRVSpellChecker 784
 - TRVSpellChecker.AddToAutoreplaceList 788
 - TRVSpellChecker.AddToDictionary 789
 - TRVSpellChecker.AddToIgnoreList 789
 - TRVSpellChecker.AvailableLanguages 786
 - TRVSpellChecker.CurrentLanguage 789
 - TRVSpellChecker.DialogShown 787
 - TRVSpellChecker.Execute 789
 - TRVSpellChecker.FillLanguageNames 790
 - TRVSpellChecker.GetAutoCorrection 790
 - TRVSpellChecker.GetLanguageNameFromCode 790
 - TRVSpellChecker.GuessCompletions 791
 - TRVSpellChecker.IsAvailable 791
 - TRVSpellChecker.IsWordValid 791
 - TRVSpellChecker.Language 787
 - TRVSpellChecker.LanguageIndex 787
 - TRVSpellChecker.OnSpellForm 793
 - TRVSpellChecker.OnSpellFormAction 794
 - TRVSpellChecker.OnSpellFormLocalize 795
 - TRVSpellChecker.RegisterAsService 791
 - TRVSpellChecker.RegisterEditor 792
 - TRVSpellChecker.SpellFormStyle 788
 - TRVSpellChecker.StartLiveSpelling 792
 - TRVSpellChecker.Suggest 792
 - TRVSpellChecker.SupportsFeatures 793
 - TRVSpellChecker.UnregisterAsService 791
 - TRVSpellChecker.UnregisterEditor 792
 - TRVSpellCheckFormStyle 788
-

- TRVSpellFormAction 1014
- TRVSpellFormActions 1014
- TRVSpellingCheckEvent 409
- TRVSTFontInfo 737
- TRVStickingItemsEvent 573
- TRVSTParaInfo 737
- TRVStringFormatMethod 1048
- TRVStringList 978
- TRVStrProperty 1026
- TRVStyle 630
- TRVStyle.AddTextStyle 657
- TRVStyle.CheckpointColor 634
- TRVStyle.CheckpointEvColor 634
- TRVStyle.Color 635
- TRVStyle.ConvertToDifferentUnits 657
- TRVStyle.ConvertToEMU 657
- TRVStyle.ConvertToPixels 657
- TRVStyle.ConvertToTwips 657
- TRVStyle.Create 657
- TRVStyle.CurrentItemColor 635
- TRVStyle.DefCodePage 635
- TRVStyle.DefTabWidth 636
- TRVStyle.DefUnicodeStyle 636
- TRVStyle.DeleteTextStyle 658
- TRVStyle.Destroy 658
- TRVStyle.DifferentUnitsToUnits 659
- TRVStyle.DisabledFontColor 637
- TRVStyle.EMUToUnits 659
- TRVStyle.EndnoteNumbering 637
- TRVStyle.FieldHighlightColor 637
- TRVStyle.FieldHighlightType 638
- TRVStyle.FindParaStyle 661
- TRVStyle.FindTextStyle 661
- TRVStyle.FloatingLineColor 638
- TRVStyle.FontQuality 639
- TRVStyle.FootnoteNumbering 637
- TRVStyle.FootnotePageReset 640
- TRVStyle.FullRedraw 640
- TRVStyle.GetAsDifferentUnits 658
- TRVStyle.GetAsEMU 658
- TRVStyle.GetAsHTMLPixels 658
- TRVStyle.GetAsPixels 658
- TRVStyle.GetAsRVUnits 658
- TRVStyle.GetAsStandardPixels 658
- TRVStyle.GetAsTwips 658
- TRVStyle.GetListStyleClass 660
- TRVStyle.GetParaStyleClass 661
- TRVStyle.GetTextStyleClass 661
- TRVStyle.GridColor 640
- TRVStyle.GridReadOnlyColor 640
- TRVStyle.GridReadOnlyStyle 640
- TRVStyle.GridStyle 640
- TRVStyle.HoverColor 641
- TRVStyle.HTMLPixelsToUnits 659
- TRVStyle.InactiveSelColor 641
- TRVStyle.InactiveSelTextColor 641
- TRVStyle.InvalidPicture 643
- TRVStyle.JumpCursor 643
- TRVStyle.LineSelectCursor 643
- TRVStyle.LineWrapMode 644
- TRVStyle.ListStyles 646
- TRVStyle.LiveSpellingColor 646
- TRVStyle.LoadINI 661
- TRVStyle.LoadReg 662
- TRVStyle.MainRVStyle 647
- TRVStyle.OnAfterApplyStyle 664
- TRVStyle.OnApplyStyle 665
- TRVStyle.OnApplyStyleColor 666
- TRVStyle.OnDrawCheckpoint 667
- TRVStyle.OnDrawPageBreak 668
- TRVStyle.OnDrawParaBack 670
- TRVStyle.OnDrawStyleText 671
- TRVStyle.OnDrawTextBack 673
- TRVStyle.OnStyleHoverSensitive 674
- TRVStyle.PageBreakColor 647
- TRVStyle.ParaStyles 648
- TRVStyle.PixelsToUnits 659
- TRVStyle.RVUnitsToUnits 659
- TRVStyle.SaveCSS 662
- TRVStyle.SaveCSSToStream 663
- TRVStyle.SaveINI 663
- TRVStyle.SaveReg 664
- TRVStyle.SelColor 641
- TRVStyle.SelectionHandleKind 648
- TRVStyle.SelectionMode 649
- TRVStyle.SelectionStyle 649
- TRVStyle.SelOpacity 641
- TRVStyle.SelTextColor 641
- TRVStyle.SidenoteNumbering 637
- TRVStyle.SoftPageBreakColor 651
- TRVStyle.SpacesInTab 652
- TRVStyle.SpecialCharactersColor 652
- TRVStyle.StandardPixelsToUnits 659
- TRVStyle.StyleTemplates 652
- TRVStyle.TextBackgroundKind 653
- TRVStyle.TextEngine 654
- TRVStyle.TextStyles 654
- TRVStyle.TwipsToUnits 659

- TRVStyle.Units 655
- TRVStyle.UnitsPixelsPerInch 655
- TRVStyle.UseSound 656
- TRVStyleConversionEvent 582, 585
- TRVStyleHoverSensitiveEvent 674
- TRVStyleLength 1027
- TRVStyleTemplate 733
- TRVStyleTemplate.ApplyToParaStyle 738
- TRVStyleTemplate.ApplyToTextStyle 739
- TRVStyleTemplate.Id 734
- TRVStyleTemplate.Kind 735
- TRVStyleTemplate.Name 735
- TRVStyleTemplate.Next 736
- TRVStyleTemplate.NextId 736
- TRVStyleTemplate.ParaStyle 737
- TRVStyleTemplate.Parent 737
- TRVStyleTemplate.ParentId 737
- TRVStyleTemplate.QuickAccess 737
- TRVStyleTemplate.TextStyle 737
- TRVStyleTemplate.UpdateModifiedParaStyleProperties 740
- TRVStyleTemplate.UpdateModifiedTextStyleProperties 740
- TRVStyleTemplate.ValidParaProperties 738
- TRVStyleTemplate.ValidTextProperties 738
- TRVStyleTemplateCollection 688
- TRVStyleTemplateCollection.AssignStyleTemplates 690
- TRVStyleTemplateCollection.AssignToStrings 690
- TRVStyleTemplateCollection.ClearParaFormat 690
- TRVStyleTemplateCollection.ClearTextFormat 690
- TRVStyleTemplateCollection.DefStyleName 688
- TRVStyleTemplateCollection.FindById 691
- TRVStyleTemplateCollection.FindByName 691
- TRVStyleTemplateCollection.FindItemById 691
- TRVStyleTemplateCollection.FindItemByName 691
- TRVStyleTemplateCollection.ForbiddenIds 689
- TRVStyleTemplateCollection.InsertFromRVST 691
- TRVStyleTemplateCollection.Items 689
- TRVStyleTemplateCollection.LoadFromRVST 692
- TRVStyleTemplateCollection.LoadFromStreamRVST 692
- TRVStyleTemplateCollection.NormalStyleTemplate 689
- TRVStyleTemplateCollection.ResetNameCounter 693
- TRVStyleTemplateCollection.SaveToRVST 692
- TRVStyleTemplateCollection.SaveToStreamRVST 692
- TRVStyleTemplateId 1027
- TRVStyleTemplateInsertMode 254
- TRVStyleTemplateKind 735
- TRVStyleTemplateName 1027
- TRVStyleUnits 1027
- TRVSubData 957
- TRVSubSuperScriptType 708
- TRVTabAlign 747
- TRVTabInfo 746
- TRVTabInfo.Align 747
- TRVTabInfo.Leader 747
- TRVTabInfo.Position 748
- TRVTabInfos 745
- TRVTabInfos.AddFrom 746
- TRVTabInfos.DeleteList 746
- TRVTabInfos.Find 746
- TRVTabInfos.Intersect 746
- TRVTabInfos.Items 745
- TRVTabItemInfo 844
- TRVTableBorderStyle 1029
- TRVTableCellData 895, 957
- TRVTableCellData.BackgroundImage 896
- TRVTableCellData.BackgroundImageFileName 897
- TRVTableCellData.BackgroundColor 897
- TRVTableCellData.BestHeight 898
- TRVTableCellData.BestWidth 898
- TRVTableCellData.BorderColor 899
- TRVTableCellData.BorderLightColor 899
- TRVTableCellData.Color 900
- TRVTableCellData.ColSpan 900
- TRVTableCellData.GetRVData 907
- TRVTableCellData.Hint 901
- TRVTableCellData.IgnoreContentHeight 901
- TRVTableCellData.Opacity 902
- TRVTableCellData.Options 902
- TRVTableCellData.properties 908
- TRVTableCellData.Rotation 904
- TRVTableCellData.RowSpan 905
- TRVTableCellData.Tag 906
- TRVTableCellData.VAlign 906
- TRVTableCellData.VisibleBorders 906
- TRVTableCellOption 902
- TRVTableCellOptions 902
- TRVTableDrawBackEvent 893
- TRVTableDrawBorderEvent 893
- TRVTableItemInfo 844, 846
- TRVTableItemInfo.AssignProperties 870
- TRVTableItemInfo.BackgroundImage 849
- TRVTableItemInfo.BackgroundImageFileName 850
- TRVTableItemInfo.BackgroundColor 850
- TRVTableItemInfo.BestWidth 850
- TRVTableItemInfo.BorderColor 851
- TRVTableItemInfo.BorderHSpacing 851
- TRVTableItemInfo.BorderLightColor 852

- TRVTableItemInfo.BorderStyle 852
- TRVTableItemInfo.BorderVSpacing 853
- TRVTableItemInfo.BorderWidth 853
- TRVTableItemInfo.CanMergeCells 870
- TRVTableItemInfo.CanMergeSelectedCells 871
- TRVTableItemInfo.CellBorderColor 853
- TRVTableItemInfo.CellBorderLightColor 854
- TRVTableItemInfo.CellBorderStyle 854
- TRVTableItemInfo.CellBorderWidth 855
- TRVTableItemInfo.CellHPadding 855
- TRVTableItemInfo.CellHSpacing 856
- TRVTableItemInfo.CellOverrideColor 856
- TRVTableItemInfo.CellPadding 856
- TRVTableItemInfo.Cells 857
- TRVTableItemInfo.CellVPadding 857
- TRVTableItemInfo.CellVSpacing 857
- TRVTableItemInfo.Changed 871
- TRVTableItemInfo.ColBandSize 858
- TRVTableItemInfo.ColCount 858
- TRVTableItemInfo.Color 858
- TRVTableItemInfo.Create 872
- TRVTableItemInfo.CreateEx 872
- TRVTableItemInfo.DeleteCols 873
- TRVTableItemInfo.DeleteEmptyCols 873
- TRVTableItemInfo.DeleteEmptyRows 873
- TRVTableItemInfo.DeleteRows 874
- TRVTableItemInfo.DeleteSelectedCols 874
- TRVTableItemInfo.DeleteSelectedRows 875
- TRVTableItemInfo.Deselect 875
- TRVTableItemInfo.Destroy 876
- TRVTableItemInfo.EditCell 876
- TRVTableItemInfo.EvenColumnsColor 867
- TRVTableItemInfo.EvenRowsColor 867
- TRVTableItemInfo.FirstColumnColor 867
- TRVTableItemInfo.GetCellAt 876
- TRVTableItemInfo.GetEditedCell 877
- TRVTableItemInfo.GetNormalizedSelectionBounds 877
- TRVTableItemInfo.GetSelectionBounds 878
- TRVTableItemInfo.HeadingRowColor 867
- TRVTableItemInfo.HeadingRowCount 859
- TRVTableItemInfo.HOutermostRule 859
- TRVTableItemInfo.HRuleColor 860
- TRVTableItemInfo.HRuleWidth 860
- TRVTableItemInfo.InsertCols 878
- TRVTableItemInfo.InsertColsLeft 879
- TRVTableItemInfo.InsertColsRight 879
- TRVTableItemInfo.InsertRows 880
- TRVTableItemInfo.InsertRowsAbove 880
- TRVTableItemInfo.InsertRowsBelow 880
- TRVTableItemInfo.IsCellSelected 881
- TRVTableItemInfo.LastColumnColor 867
- TRVTableItemInfo.LastRowColor 867
- TRVTableItemInfo.LoadFromStream 881
- TRVTableItemInfo.MergeCells 881
- TRVTableItemInfo.MergeSelectedCells 882
- TRVTableItemInfo.MoveRows 882
- TRVTableItemInfo.OddColumnsColor 867
- TRVTableItemInfo.OddRowsColor 867
- TRVTableItemInfo.OnCellEditing 892
- TRVTableItemInfo.OnDrawBackground 893
- TRVTableItemInfo.OnDrawBorder 893
- TRVTableItemInfo.Opacity 860
- TRVTableItemInfo.Options 861
- TRVTableItemInfo.PrintOptions 863
- TRVTableItemInfo.RowBandSize 858
- TRVTableItemInfo.RowCount 864
- TRVTableItemInfo.Rows 864
- TRVTableItemInfo.SaveRowsToStream 883
- TRVTableItemInfo.SaveToStream 883
- TRVTableItemInfo.Select 884
- TRVTableItemInfo.SelectCols 884
- TRVTableItemInfo.SelectRows 884
- TRVTableItemInfo.SetCellBackgroundImage 885
- TRVTableItemInfo.SetCellBackgroundImageFileName 885
- TRVTableItemInfo.SetCellBackgroundStyle 885
- TRVTableItemInfo.SetCellBestHeight 886
- TRVTableItemInfo.SetCellBestWidth 886
- TRVTableItemInfo.SetCellBorderColor 886
- TRVTableItemInfo.SetCellBorderLightColor 886
- TRVTableItemInfo.SetCellColor 887
- TRVTableItemInfo.SetCellHint 887
- TRVTableItemInfo.SetCellOpacity 887
- TRVTableItemInfo.SetCellOptions 887
- TRVTableItemInfo.SetCellRotation 888
- TRVTableItemInfo.SetCellTag 888
- TRVTableItemInfo.SetCellVAlign 888
- TRVTableItemInfo.SetCellVisibleBorders 889
- TRVTableItemInfo.SetRowKeepTogether 889
- TRVTableItemInfo.SetRowPageBreakBefore 889
- TRVTableItemInfo.SetRowVAlign 889
- TRVTableItemInfo.SetTableVisibleBorders 889
- TRVTableItemInfo.SplitSelectedCellsHorizontally 890
- TRVTableItemInfo.SplitSelectedCellsVertically 890
- TRVTableItemInfo.TextColSeparator 865
- TRVTableItemInfo.TextRowSeparator 865
- TRVTableItemInfo.UnmergeCells 890
- TRVTableItemInfo.UnmergeSelectedCells 891

TRVTableItemInfo.VisibleBorders 865
 TRVTableItemInfo.VOutermostRule 866
 TRVTableItemInfo.VRuleColor 866
 TRVTableItemInfo.VRuleWidth 867
 TRVTableOption 861
 TRVTableOptions 861
 TRVTablePrintOption 863
 TRVTablePrintOptions 863
 TRVTableRow 909
 TRVTableRow.Items 910
 TRVTableRow.KeepTogether 910
 TRVTableRow.PageBreakBefore 910
 TRVTableRow.VAlign 910
 TRVTableRows 911
 TRVTableRows.GetMainCell 912
 TRVTableRows.Items 912
 TRVTabNavigationType 255
 TRVTag 1029
 TRVTextBackgroundKind 653
 TRVTextBoxItemInfo 947
 TRVTextBoxItemInfo.Create 948
 TRVTextBoxItemInfo.CreateEx 948
 TRVTextDrawState 1030
 TRVTextDrawStates 1030
 TRVTextEngine 654
 TRVTextFoundEvent 410
 TRVTextItemInfo 844
 TRVTextOption 704
 TRVTextOptions 704
 TRVThumbnailMaker 979
 TRVInplaceRVData 957
 TRVUnderlineType 709
 TRVUndoType 1031
 TRVUnicodeString 1032
 TRVUnicodeTestResult 992
 TRVUnits 1032
 TRVUnitsRect 980
 TRVVAAlign 1033
 TRVVerticalAnchor 945
 TRVVirtualPrinterProperties 783
 TRVVirtualPrinterProperties.Active 784
 TRVVirtualPrinterProperties.PageHeight 784
 TRVVirtualPrinterProperties.PageWidth 784
 TRVWriteHyperlink 411
 TRVWriteObjectPropertiesEvent 413
 TRVYesNoAuto 1034
 TRVZoomMode 840
 TRVZoomPercent 1035
 TwipsToUnits 659

TRVStyle 659

- U -

UnderlineColor 709
 TCustomRVFontInfo 709
 UnderlineType 709
 TCustomRVFontInfo 709
 Undo 567
 TRichViewEdit 567
 Undo in RichViewEdit 202
 Undo in TRichViewEdit 115
 Undo/redo in Tables 202
 UndoAction 567
 TRichViewEdit 567
 UndoLimit 481
 TRichViewEdit 481
 UndoName 568
 TRichViewEdit 568
 Unicode 130, 715
 TFontInfo 715
 Unicode in TRichView 130
 UnicodeMode 459
 TRVRTFReaderProperties 459
 Units 420, 655, 770
 TRVDocParameters 420
 TRVPrint 770
 TRVStyle 655
 Units of measurement 139
 UnitsPerInch 421
 TRVDocParameters 421
 UnitsPixelsPerInch 655
 TRVStyle 655
 UnmergeCells 890
 TRVTableItemInfo 890
 UnmergeSelectedCells 891
 TRVTableItemInfo 891
 UnregisterAsService 791
 TRVSpellChecker 791
 UnregisterEditor 792
 TRVSpellChecker 792
 UpdateModifiedParaStyleProperties 740
 TRVStyleTemplate 740
 UpdateModifiedTextStyleProperties 740
 TRVStyleTemplate 740
 UpdatePageNumbers 810
 TRVReportHelper 810
 UpdatePaletteInfo 368, 828
 TCustomRVPrint 828
 TRichView 368

URLFormatName 1037
 UseCharsetForUnicode 459
 TRVRTFReaderProperties 459
 UseCheckpointNames 437
 TRVHTMLSaveProperties 437
 UseFMXThemes 256
 TRichView 256
 UseHypertextStyles 459
 TRVRTFReaderProperties 459
 UseSingleCellPadding 460
 TRVRTFReaderProperties 460
 UseSound 656
 TRVStyle 656
 UseStyleTemplates 256, 977
 TRichView 256
 TRVReportHelperWithHeaderFooters 977
 UseVCLThemes 257
 TRichView 257
 UseXPThemes 818
 TRVScroller 818

- V -

Valid documents 138
 Valid TRichView documents 138
 ValidParaProperties 738
 TRVStyleTemplate 738
 ValidTextProperties 738
 TRVStyleTemplate 738
 VAlign 257, 906, 910, 939
 TRichView 257
 TRVBoxProperties 939
 TRVTableCellData 906
 TRVTableRow 910
 VerticalAlignment 946
 TRVBoxPosition 946
 VerticalAnchor 945
 TRVBoxPosition 945
 VerticalOffset 946
 TRVBoxPosition 946
 VerticalPositionKind 946
 TRVBoxPosition 946
 Viewer vs Editor 135
 VirtualPrinter 770
 TRVPrint 770
 VisibleBorders 744, 865, 906
 TRVBorder 744
 TRVTableCellData 906
 TRVTableItemInfo 865
 VOutermostRule 866

 TRVTableItemInfo 866
 VRuleColor 866
 TRVTableItemInfo 866
 VRuleWidth 867
 TRVTableItemInfo 867
 VScrollMax 818
 TRVScroller 818
 VScrollPos 819
 TRVScroller 819
 VScrollVisible 819
 TRVScroller 819
 VShift 710
 TCustomRVFontInfo 710
 VSmallStep 257
 TRichView 257

- W -

WheelStep 819
 TRVScroller 819
 Width 744, 939
 TRVBorder 744
 TRVBoxProperties 939
 WidthType 939
 TRVBoxProperties 939
 WordWrap 258
 TRichView 258

- X -

XHTMLCheck 428
 TRVHTMLReaderProperties 428

- Z -

ZoomInCursor 839
 TCustomRVPrintPreview 839
 ZoomMode 420, 840
 TCustomRVPrintPreview 840
 TRVDocParameters 420
 ZoomOutCursor 840
 TCustomRVPrintPreview 840
 ZoomPercent 258, 421, 841
 TCustomRVPrintPreview 841
 TRichView 258
 TRVDocParameters 421