

---

**Programmer's Guide**

# **TList 8 ActiveX Control**

**Bennet-Tec Information Systems, Inc.**

This manual was produced using *ComponentOne Doc-To-Help*.

# Contents

Bennet-Tec Information Systems, Inc. ....	1
Software License Agreement: Use and Distribution. ....	1
List of Custom Controls ....	iii
Contact Information ....	iv
Introduction .....	5
Control features.....	5
New Features (Version 8.0).....	8
License Registration.....	8
License Registration Questions and Answers.....	10
On-Line Help .....	12
Distribution Notes .....	12
Technical Support .....	14
Features and programming techniques .....	15
TList features and programming techniques.....	15
Backward Compatibility.....	16
Color Support.....	17
Design-Time Support .....	18
Display Features.....	18
Expanding and Collapsing the Outline.....	20
Hiding TList Items .....	20
Grid Support.....	21
Internet Interfaces .....	22
Keyboard Interface.....	22
Navigating the List - Choosing an Indexing Scheme.....	24
Objects and Object Collections.....	27
Picture, Selection and Double-Byte Characters Support .....	29
Visual Elements and Hot Spots.....	31
How to Add or Delete Items.....	32
How to Boost Performance.....	33
How to Specify Default Properties.....	34
How to Specify and Work with Associated Hidden Data.....	35
How to Work with Virtual Items .....	36
How to Use TList Grids for Column/Table Data .....	37
How to Work with Databases.....	43
How to Use Tree Buffers to Manipulate the Tree .....	46
How to Support DragDrop .....	47
How to Sort or Search a Tree .....	51
How to Sort a Grid.....	51
How to Access the Clipboard.....	52
How to Save and Load Lists - File I/O .....	53
How to Use Cell / Item Editing .....	54
How to Use Bookmarks.....	65
How to Assign Categories - TList Mark Support.....	66
How to Control the Display of Plus/Minus Pictures.....	66
How to Upgrade an Old TList 3/Pro OCX Project to Use New TList OCX .....	67
How to Upgrade an Old VBX Based Project to Use TList OCX in Place of the VBX .....	67
How To Detect the Version Number Of TList.....	69
How to Trap Right Mouse Clicks.....	69

How to Navigate a Web Site with TList.....	69
How to Print With TList.....	70
How to Use TList's RTF Support.....	72
How to Use the VisualRoot Property.....	72
How to Use LevelDefs for Sorting.....	73
How to Use TListNode and TListNode Objects.....	74
How to Use IntelliMouse Functionality.....	74
How To Resize Rows And Columns.....	75
How to Use ToolTips.....	75
How to Copy from one TList to Another.....	76
How to Smoothly Scroll Tall Items.....	76
How to Support Excel Style Navigation While Editing.....	76
TList Diagrams (Objects, Functionality And Relationships).....	77
TListTreeView features and programming techniques.....	83
MSTreeView and TListTreeView compatibility.....	83
TListTreeView and TList.....	84
Using the TDesigner application.....	88
TDesigner Layout.....	89
Design-Time Mode buttons.....	90
Using TDesigner Windows.....	92
Specifying TDesigner Defaults.....	93
Using Clipboard.....	93
Using Drag/Drop.....	94
Printing.....	95
Hints for Web Site Designers.....	96
Setting up ToolTips.....	99
Selecting Colors.....	99
Setting up Pictures.....	102
Modifying Tree Line Settings.....	105
Specifying Drag Drop Settings.....	106
Controlling Selection.....	106
Controlling Expanding/Collapsing.....	107
Controlling Text Display.....	108
Controlling Fonts.....	110
Controlling Marks.....	111
Associating Additional Data with an Item.....	112
Specifying Sorting Method.....	113
Controlling Miscellaneous Settings.....	114
Controlling Item Cell Default Settings.....	115
Setting up Background.....	116
Specifying Scrollbar Appearance.....	117
Setting up Item And Grid Cell Borders.....	118
Setting up Item And Grid Cell Alignment.....	118
Specifying Virtual Items.....	118
Setting up Item Visibility.....	118
Setting up Item And Grid Cell Tag.....	119
Setting up LevelDefs.....	119
Specifying a Tree Grid.....	119
Specifying Item Grids.....	119
Properties You Cannot Set with TDesigner.....	119
Objects reference.....	121
Introduction.....	121
TList Object.....	123
TListCellDef Object.....	137
TListCheckBox Object.....	139
TListColDef Object.....	140
TListColDefs Object Collection.....	141
TListComboBox Object.....	141
TListComboItem Object.....	142

TListComboItems Object .....	144
TListDataObject Object.....	145
TListDataObjectFiles Object .....	146
TListDateTime Object.....	147
TListEditInfo Object .....	148
TListEditingChangeInfo Object.....	148
TListGrid Object .....	149
TListGridCell Object.....	152
TListLevelDef Object.....	153
TListLevelDefs Object Collection .....	154
TListNode Object.....	154
TListNodes Object Collection .....	156
TListReport Object.....	156
TListPage Object.....	158
TListPages Object Collection .....	159
TListSpin Object .....	159
TListTextBox Object.....	160
TListValue Object.....	161
TListValues Object Collection .....	162
TListSelectedGridCells Object .....	162
TListSelectedGridColumns Object .....	163
TListSelectedGridRows Object .....	163
TListRowDefs Object.....	164
TListRowDef Object .....	164
TVWSelectedNodes Object Collection (TListTreeView).....	165
TListPoint Object.....	166
TListRectangle Object.....	166
TListShowTooltipArgs Object.....	167
TListTooltip Object.....	167
TListTooltipStyle Object.....	168
TListTooltipWindow Object.....	169
Properties, events, methods, functions reference .....	171
AbortWindowStyle and AbortWindow Properties.....	171
About Property.....	171
Activatable Property (TListCellDef Object).....	172
Activate Method (TListGrid object).....	173
ActivationMode Property (TListGrid object) .....	174
ActiveGrid Property .....	176
ActiveCell Property (TListGrid object).....	176
ActiveRow Property (TListGrid object).....	177
Add and SafeAdd Methods (TListComboItems Object) .....	177
Add Method (TListSelectedGridCells, TListSelectedGridColumns and TListSelectedGridRows Objects) .....	178
Add Property .....	179
Add Method (TListNode/TlistNodes objects) .....	180
Add Method (TVWSelectedNodes Object) .....	181
AddAfter Method .....	181
AddItem Method .....	182
AddItem2 and AddItem2Ex Methods .....	184
AddRow Method.....	185
AfterEditing Event .....	186

Align Property .....	188
AllowResizing Property .....	188
Appearance Property .....	189
Appearance Properties (TListCheckbox Object) .....	189
AutoDragComplete Event .....	190
AutoDragRequest Event .....	191
AutoDragMode Property .....	192
AutoExpand Property .....	192
AutoFillColTitles and AutoFillRowTitles Properties .....	193
AutoNewPage Property .....	194
AutoScrDuringDragDrop Property .....	194
AutoSizeRow Method, AutoSizeColumn Method .....	195
AutoSizeOptions Property .....	196
BackColor and DefItemCellBackColor Property .....	197
BackColor and SelBackColor Properties (TListNode Object) .....	198
BackColorBkg Property .....	198
BackPicture and BackPictureAlignment Properties .....	198
BackwardCompatible Property .....	199
BeforeDrag Method .....	199
BeforeItemDeactivate, <i>BeforeGridRowDeactivate and</i> <i>BeforeGridCellDeactivate</i> Events .....	200
BeginPage Event .....	203
BorderColor and DefItemCellBorderColor Properties .....	203
BorderStyle and DefItemCellBorderStyle Properties .....	204
BorderStyle Property (TListGrid Object) .....	205
BottomIndex Property .....	206
Caption Property .....	206
CellDef Property .....	207
CellEdit Property .....	207
Cells Property .....	208
CellValue Property (TListComboItem Object) .....	208
CheckBox Property (TListEditInfo Object) .....	209
CheckboxValue Property .....	210
CheckedPicture, UncheckedPicture and GrayedPicture Properties (TListCheckbox Object) .....	210
CheckedValue, UncheckedValue and GrayedValue Properties (TListCheckbox Object) .....	211
Children Property (TListNode Object) .....	212
ChildrenCount Property (TListNode Object) .....	213
Col and Row Properties (TListGridCell object) .....	213
Col and Row Properties (TListGrid object) .....	213
ColDefs Property .....	214
ColDelimiter Property .....	214
Cols and Rows Properties .....	215
ColTitleCellDef Property .....	215
ColTitlesHeight Property .....	215
ComboBox Property (TListEditInfo Object) .....	216
ConvertTabsToCols Property .....	216
Count Property .....	216
Count Property (TListSelectedGridCells, TListSelectedGridColumns and TListSelectedGridRows Objects) .....	217

Count Property (TVWSelectedNodes Object) .....	217
Clear Method .....	217
Clear Method (TListNode Object) .....	218
Clear Method (TListSelectedGridCells, TListSelectedGridColumns and TListSelectedGridRows Objects) .....	218
Clear Method (TListComboItems Object) .....	219
Clear Method (TVWSelectedNodes Object) .....	219
ClearItem Property .....	219
Click Event .....	220
Clipboard Property .....	220
CoerceIndex Property .....	221
Collapse Event .....	221
CopyBuffer Method .....	222
CopyItem Property .....	222
CopyItemSub Property .....	222
CopyOne Property .....	223
CopySelected Property .....	223
CurrentIndexMethod Property .....	224
CurrentItem Property .....	225
CurrentParent Property .....	225
CurrentItemBM Property .....	226
DateTime Property (TListEditInfo Object) .....	226
DefItemCellAlignment and Alignment Properties .....	227
DefItemCellPictureAlignment and PictureAlignment Properties .....	228
DefItemCellTextAlignment and TextAlignment Properties .....	229
DblClick Event .....	229
DefItemCellDef Property .....	230
DefMultiLine Property .....	230
DisableNoScroll Property .....	230
DisplayValue Property (TListComboItem Object) .....	231
DisplayPic Property (TListComboItem Object) .....	232
Drag Method .....	233
DragColumnsEnabled property .....	233
DragDrop, DragOver Events .....	233
DragDropEx, DragOverEx Events .....	234
DragHighlight Property .....	234
DragIcon Property .....	235
DragIconStyle Property .....	235
DragMode Property .....	236
DrawFocusRect Property .....	236
DropDownItemHeight Property (TListComboBox) .....	237
DropDownMaxHeight Property (TListComboBox) .....	237
DropDownWidth Property (TListComboBox) .....	238
DropTarget Property .....	238
EditAreaMinHeight, EditAreaMaxHeight, EditAreaMinWidth, EditAreaMaxWidth Properties (TListComboBox) .....	239
Editable Property (TListEditInfo) .....	239
EditableStartOptions Property (TListEditInfo) .....	240
EditInfo Property .....	241
EditingKeyDown Event .....	242
EditingKeyPress Event .....	242

EditingKeyUp Event .....	242
EditingMode Property .....	242
EditInfoObject Property (TListEditingChangeInfo object) .....	243
Enabled Property .....	244
EndPage Event .....	244
EnumIndex Property (TListNode Object) .....	245
Environment Property .....	245
Expand Event .....	246
Expand Property .....	246
ExpandChildren Property .....	247
ExpandEx Property .....	247
ExpandNewItem Property .....	248
ExpandToLevel Property .....	248
ExplorerCompatible Property .....	249
FastAddItem and FastAddItemEx Methods .....	249
File Property .....	249
Files Property .....	250
FindItem and FindValue Methods .....	250
FireTListEvents Property (TListTreeView) .....	251
FirstItem and LastItem Properties .....	252
FirstSibling and LastSibling Properties (TListNode Object) .....	252
FixedSize Property .....	252
FocusRectStyle property .....	253
Font Property .....	253
FontBold, FontItalic, FontStrikethru, FontUnderline Properties .....	254
FontName Property .....	255
FontSize Property .....	255
Font3D property (TListCellDef object) .....	255
FontShadowColor and FontShadowSelectedColor properties (TListCellDef object) .....	256
ForeColor Property .....	257
Format Property .....	257
Format Property (TListDateTime Object) .....	262
FormatString Property (TListDateTime Object) .....	264
FreeBuffer Method .....	266
FreeBuffer/TListFreeBuffer improvement .....	266
FullItemString and FullRowString Properties .....	266
FullPath Property .....	267
GetArrayProperty, SetArrayProperty, GetArrayPropertyID Properties .....	267
GetData Method .....	268
GetFormat Method .....	269
GetItemByCellValue Method (TListComboItems) .....	269
GetItemByXY Method .....	270
GetItemRect Method .....	271
GotFocus Event .....	271
GradientColorFrom, GradientColorTo, and GradientStyle Properties .....	272
Grid Property .....	273
GridCellActivate Event / GridCellDeactivate Event .....	273
GridCellClick Event .....	274
GridCellDbClick Event .....	275



GridCellDef Property .....	275
GridCellRequestEditing Event.....	276
GridCellAfterEditing Event.....	277
GridCellEditingChange Event .....	278
GridCellEditingKeyDown, GridCellEditingKeyUp and GridCellEditingKeyPress Events .....	279
GridLinesColor Property .....	280
GridLinesStyle Property .....	280
GridLines3DStyle Property .....	281
GridLines3DLightColor and GridLines3DShadowColor Properties .....	281
GridRowActivate Event / GridRowDeactivate Event .....	282
GridRowTitleClick Event, GridColumnTitleClick Event, GridCornerTitleClick Event .....	282
HasCell Property .....	283
HasGrid Property .....	284
HasSubItems Property .....	284
Height Property .....	285
HelpContextID Property.....	285
HScroll and VScroll Events.....	285
HWnd Property .....	286
Image Property.....	287
ImageStretch Property .....	288
Indent Property.....	288
Indentation Property .....	289
Index Property (TListNode Object) .....	290
Index Property.....	290
IndexByBM Method .....	291
Insert Property.....	291
InsertItem Property.....	292
InvBorderStyle Property.....	292
InvImage Property or SelectedImage Property .....	293
InvStyle Property .....	294
IsClipboardAvailable Property .....	294
IsItemVisible Property .....	295
IsPrinting Method .....	295
IsValidBM Method .....	295
IsValidBuffer Method .....	295
Item Method (TListSelectedGridCells, TListSelectedGridColumns and TListSelectedGridRows Objects) .....	296
Item Property (TVWSelectedNodes Object) .....	297
ItemActivate Event / ItemDeactivate Event .....	297
ItemAlwaysHidden Property .....	298
ItemBackColor and ItemForeColor Properties .....	298
ItemBM Property .....	299
ItemCell Property .....	299
ItemCheckboxValue Property.....	300
ItemClick Event .....	301
ItemDbClick Event.....	301
ItemEditText Property.....	301
ItemEditingChange Event.....	302
ItemFont... Properties .....	303

ItemGrid Property .....	304
ItemIndex Property.....	305
ItemIndexToRow Method .....	305
ItemHasGrid Property .....	305
ItemHeight Property.....	306
ItemImageDefWidth and ItemImageDefHeight Property .....	306
ItemMark Property .....	307
ItemMultiLine Property.....	307
ItemParent Property.....	308
ItemParentBM Property.....	308
ItemPMPicType Property .....	308
ItemPrevSibling, ItemNextSibling, and ItemLastSubItemIndex Properties .....	309
Item Property .....	309
ItemQueryData Event.....	310
Items Property (TListComboBox Object) .....	311
ItemSorted, ItemSortingMethod, ItemSortingStyle, and ItemSortingKey Properties .....	311
ItemTag Property .....	315
ItemToolTip and ToolTip Properties .....	316
Item... Value Properties .....	317
ItemValues and ItemHasValue Properties (Values property of TListNode Object) .....	317
ItemVirtualParent, ItemVirtualCount, VirtualParent and VirtualCount Properties.....	318
ItemUrl Property .....	319
HitTest Method .....	319
KeyboardActivation Property (TList object).....	320
KeyDown and KeyUp Events .....	322
KeyPress Event .....	322
KeyboardNavigateWhileEditing Property(TListEditInfo) .....	322
Left Property .....	323
LeftMargin, TopMargin, RightMargin, and BottomMargin Properties .....	324
LevelDefs Property .....	324
List Property .....	324
ListCount Property .....	325
ListCountEx Property.....	325
ListIndex Property.....	326
LoadAndAdd Property .....	326
LoadAndInsert Property .....	327
LoadBuffer Method.....	327
LoadData Method .....	328
LoadData Method (TListTreeView) .....	328
LoadPicture Method.....	329
LostFocus Event.....	329
MarginLeft, MarginTop, MarginRight, MarginBottom Properties.....	330
MarkClick and MarkDbClick Events.....	330
MarkedItemsAlwaysHidden Property .....	330
MarkPicture Property .....	331
MarkTag Property .....	331
MarkWidth and MarkHeight Properties .....	332

MaxLength Property (TListTextBox Object) .....	332
Min and Max Properties (TListDateTime Object) .....	333
Min and Max Properties (TListSpin Object) .....	333
MinHeight, MaxHeight and HeightScale Properties (TListTextBox Object) .....	333
MinWidth and MaxWidth Properties (TListTextBox Object) .....	334
Modifications Property .....	335
MoveItem Method .....	340
MoveTo Method .....	342
MouseCol and MouseRow Properties .....	342
MouseDown and MouseUp Events .....	343
MouseMove Event .....	343
MousePointer Property .....	343
MouseIcon Property .....	344
MouseWheel Event .....	344
Move Method .....	345
MSOutlineAdd Property .....	345
MS TreeView Standard Property/Method/Event .....	345
MultiLine Property .....	345
MultiSelect Property .....	346
Name Property .....	347
NewIndex Property .....	347
Next and Prev Properties (TListNode Object) .....	348
Node Property .....	348
NodeFromItem, NodeFromGridCell, NodeFromTListNode Methods (TListTreeView object) .....	348
NoIntegralHeight Property .....	349
NonScrollableColumns Property .....	349
NoPictureRoot Property .....	349
OldSelStart And OldSelLength Properties (TListEditingChangeInfo object) .....	350
OLECompleteDrag event .....	351
OLEDrag method .....	351
OLEDragMode property .....	352
OLEDragDrop Event .....	353
OLEDropMode property .....	354
OLEDragOver Event .....	355
OLEGiveFeedback event .....	356
OLEStartDrag event .....	357
OLESetData event .....	358
OnDragDrop and OnDragOver Methods .....	359
Options Property (TListCheckbox Object) .....	359
Options Property (TListDateTime Object) .....	360
Options Property (TListSpin Object) .....	361
Options Property (TListTextBox Object) .....	362
Options Property (TListComboBox Object) .....	363
Pages Property .....	364
Parent Property (TListNode Object) .....	365
PasteBuffer Method .....	365
Parent Property .....	366
ParentItemIndex Property .....	366

PathSeparator Property .....	366
Picture and PictureSelected Properties .....	367
Picture... Properties .....	367
PicturePalette Property .....	368
PicInMultiLine Property .....	368
PictureClick Event .....	369
PictureDblClick Event .....	369
PictureMark Property .....	369
PicturePlus and PictureMinus Properties .....	369
PictureType Property .....	370
PictureWidth and PictureHeight Properties .....	370
PlusMinusClick and PlusMinusDblClick Events .....	371
PostScriptDC Property .....	371
PrepareForPrinting Method .....	372
PreparePage Event .....	372
PreviewMode Property .....	373
PreviewPageWidth and PreviewPageHeight Properties .....	373
PrintBackground Property .....	373
PrintLevels Property .....	374
PrinterObject Property .....	374
PrintingStop Method .....	374
PrintOneStep Method .....	375
RecalculateSize Method (TListTooltipWindow Object) .....	376
Redraw Property .....	377
Redraw Property (TListTreeView) .....	377
Refresh Method .....	378
RefreshItems Method .....	378
Remove Method (TListComboItems Object) .....	378
Remove Method (TListNodes Object) .....	379
Remove Method (TListSelectedGridCells, TListSelectedGridColumn and TListSelectedGridRows Objects) .....	379
Remove Method (TVWSelectedNodes Object) .....	380
RemoveItem Method .....	381
RemoveItem Method (TListComboItems Object) .....	381
RemoveRow Method .....	382
Report Property .....	382
RequestEditing Event .....	382
Root Property .....	384
RowCellDef Property .....	384
RowDefs Property (TListGrid object) .....	384
RowHeight Property .....	384
RowTitleCellDef Property .....	385
RowTitlesWidth Property .....	385
RowToItemIndex Method .....	386
RTFStyle Property .....	386
Save Property .....	386
SaveBuffer Method .....	387
SaveData Method .....	387
SaveData Method (TListTreeView) .....	388
SaveOne Property .....	388
SaveSub Property .....	389

Scrollbars Property .....	389
ScrollHorz Property .....	390
ScrollHPosition property .....	390
ScrollHRange property .....	390
ScrollVModeMouse, ScrollVModeKeyboard, ScrollVStepMouse and ScrollVStepKeyboard properties .....	391
ScrollVPosition property .....	392
ScrollVRange property .....	393
SelBackColor and SelForeColor Properties .....	393
SelBorderStyle property (TListCellDef object) .....	393
SelBorderColor property (TListCellDef object) .....	394
Selectable Property (TListCellDef object) .....	395
Selected Property .....	395
Selected Property (TListGridCell, TListRowDef and TListColDef objects) .....	396
SelectedCells Property (TListGrid object) .....	397
SelectedColumns Property (TListGrid object) .....	397
SelectedRows Property (TListGrid object) .....	397
SelectedItem Property (TListTreeView) .....	397
SelectedNodes Property (TListTreeView) .....	398
SelectEx Property .....	398
SelectionMode property (TListGrid object) .....	399
SelectionOptions Property (TListGrid object) .....	401
SelectionMode Property (TListTreeView) .....	402
SelItemCount Property .....	403
SelItemIndex Property .....	403
SelStart And SelLength Properties (TListEditingChangeInfo object) .....	403
SetFocus Method .....	404
Shift Property .....	405
ShiftStep Property .....	405
ShowCaption Property .....	406
ShowChildren Property .....	406
ShowHiddenItems Property .....	407
ShowColumnTitles Property .....	407
ShowColTitles Property .....	407
ShowRowTitles Property .....	408
ShowTitles Property .....	408
SmartDragDrop Property .....	408
ShowTooltip Event .....	409
Sorting Property (TListComboItems Object) .....	410
SortingKey Property (TListComboItems Object) .....	411
SortingStyle Property (TListComboItems Object) .....	411
Sorted, SortingMethod, SortingStyle, and SortingKey Properties .....	412
Spin Property (TListEditInfo Object) .....	416
Start Method .....	416
Step Property (TListSpin Object) .....	416
Style Property (TListEditInfo Object) .....	417
States Property (TListCheckBox Object) .....	418
Style Property (TListComboBox Object) .....	418
TabIndex Property .....	419
TabStop Property .....	419

TabStopDistance Property .....	420
Tag Property .....	420
Text Property .....	420
TextBox Property (TListEditInfo Object) .....	421
TitleHeight Property.....	421
TitlePicture Property .....	421
TitleText Property .....	421
TitleVisible Property .....	422
TitleWidth Property.....	422
TitlesResize Event.....	422
TList Property (TListTreeView) .....	423
TListCopyBuffer Function .....	423
TListFind ... Functions .....	423
TListFreeBuffer Function.....	424
TListGetItemByXY Function .....	424
TListGetItemRect Function .....	424
TListIndexByBM Function .....	425
TListIsClipboardFormatAvailable Function.....	425
TListIsValidBM Function .....	425
TListIsValidBuffer Function .....	425
TListLoadBuffer Function.....	426
TListPasteBuffer Function.....	426
TListSaveBuffer Function .....	426
TListTranslateIndex Function.....	426
TListNode Property (TListTreeView) .....	427
ToolTipsBackColor Property.....	427
ToolTipsDelay Property .....	427
ToolTipsForeColor Property.....	428
ToolTipsMode Property .....	428
ToolTipsText Property .....	428
ToolTipsViewStyle Property .....	429
Top Property .....	429
TopIndex Property .....	430
TranslateIndex Method.....	430
TransparentBackground Property .....	430
TransparentBitmap Property.....	431
TransparentBitmapColor Property .....	431
TreeGrid Property .....	431
TreeLinesColor Property .....	432
TreeLinesStyle Property .....	432
TreeLinesColor, TreeLinesHighlightColor and TreeLinesShadowColor properties.....	433
TriggerEvents Property .....	434
UpdateBackground Method.....	435
UpdateImages Method (TListTreeView ) .....	435
Url Property .....	435
Value Property .....	435
ValueName Property .....	436
Value and OldValue Properties (TListEditingChangeInfo object) .....	436
ValueType Property (TListEditingChangeInfo object) .....	437
Version Property .....	439

ViewStyle Property .....	439
ViewStyleEx Property .....	440
Visible Property .....	440
Visible Property (TListNode Object) .....	441
VisualRoot Property .....	441
WebAutoNavigate Property .....	441
WebTargetFrame Property .....	442
WebURLBase Property .....	442
WebGoBack Method .....	442
WebGoForward Method .....	442
WebNavigate Method .....	443
WheelScrolling Property .....	443
Width Property .....	444
WidthOfText Property .....	445
WidthOfTextMin Property .....	445
XOffset Property .....	446
Zoom Property .....	446
ZOrder Method .....	446
Error messages .....	447
Trappable Errors .....	447
Index .....	451





# Bennet-Tec Information Systems, Inc.

---

## Software License Agreement: Use and Distribution.

This is a legal agreement governing the use of the TList™ custom control (file TList8.OCX) hereafter referred to as "TList" or "TList Control", as well as the use of any associated files distributed by Bennet-Tec Information Systems with TLIST (such files including TList are hereafter collectively referred to as "TList Package". Terms of this agreement are binding upon any individual using TList, the employer of any such individual using TList under the terms of his/her employment, and any individual or organization purchasing the TList Package, and any software publisher distributing an application built using TList.

**1. Ownership:** TList and the TList Package are owned by Bennet-Tec Information Systems, Inc. and are protected by US copyright laws and international treaty provisions. Neither the TList Package nor any accompanying documentation may be copied for distribution or resale, in whole or in part, without express permission from Bennet-Tec, except as stipulated below (see Distribution of Run Time Software).

**2. Grant of License.** This license agreement permits a single individual, to use TList in creating compiled applications. This license **MAY NOT BE SHARED OR TRANSFERED**. A separate license must be purchased for each individual loading TList in a design time environment. To validate the license, each individual must complete and return a copy of the registration /feedback form.

**3. Restrictions on Use -** Use of the TList for purposes of reverse engineering is expressly prohibited. TList may not be used to create other custom controls or software components (such as OCX/ActiveX or DLL's) for use in a software design environment, except as such derived software is designed to require the purchase of a TList license from Bennet-Tec in order to be used in a design environment. Applications created with TList may not be distributed prior to completion and return of the registration/feedback form and notification to Bennet-Tec of the distribution.

**4. Distribution of Run Time software:** Subject to restrictions identified above, the file "TLIST8.OCX" may be distributed without any additional fees or royalties with any compiled application, created with TList by a licensed individual, which does NOT itself duplicate the functionality of the control within a Design time environment. Neither the help file TLIST8.HLP, nor the design-time support file TDESIGN8.EXE may be distributed under any circumstances - these files are for use solely by the licensed user of the TList control. In distributing TList based applications, the appropriate OCX file should be copied into the end-user's windows/System or System32 directory. Documentation (on-line files or hard copy) distributed with TList based applications should clearly identify Bennet-Tec Information Systems, Inc. as the copyright owner for the TList control. Application publishers must inform Bennet-Tec of the application title for any distributed application making use of the TList control, as well as the name(s) of the licensed individual(s) having built such application.

**5. Limited Warranty:** Bennet-Tec warrants that the software (TList) will perform as advertised and as provided for in the documentation for a period of ninety (90) days from the date of receipt. Should the software (TList) fail to perform as advertised within such a period, Bennet-Tec will make every reasonable effort to satisfy any such claims, alternatively the customer may return the software (original disks and documentation) within this period with proof of purchase for a full refund of the purchase price (not to include shipping and handling charges).

**6. Other Warranties:** To the maximum extent permitted by law, Bennet-Tec disclaims all other warranties, expressed or implied.

**7. No Liability for Consequential Damages:** To the maximum extent permitted by law, Bennet-Tec will refuse to accept any liability for damages whatsoever arising out of the use or inability to use this product (TList), even if Bennet-Tec has been advised of the possibility of such damages. Individuals, their employers, and application publishers making use of the TList control are fully responsible for testing applications built using the software and accept responsibility for any liability or damages incurred by the end-user of such applications resulting from the use or inability to use such applications.

**8. Contact Information:** any questions regarding this agreement or concerning TList itself should be directed to us directly through the Support page of the Bennet-Tec web site.

---

## List of Custom Controls



**ALLText** is a multiple font text box with full format editing and/or write protected presentation: supports paragraph formatting, multiple fonts, font characteristics, colors, formatted cut & paste (even between applications), transparent backgrounds, automated I/O and much more. Breaks the 32 K barrier. Easy to use - many standard properties and events.

ALLText HT/Pro is an enhanced Rich Text Box control, includes all features of our basic ALLText control plus Hypertext tags, Bookmark tags, RTF File support, Embedded Pictures (WMF, BMP, ICO, JPEG) and OLE Objects. Data Aware under Visual Basic, Hidden Text, and more.

ALLText HTML includes all the features of ALLText HT/Pro plus Read/Write/Edit of Level 2 HTML plus Tables. Thus you get TXT, RTF and HTML support in one control.



**MetaDraw** is the Object Oriented Graphics control. Ideal for Diagramming, Drawing / CAD , Image Annotation, Merging Images, Hypergraphic HotSpots, User Interface Design, and much more.

MetaDraw supports creation, display and manipulation of graphic objects within a picturebox – Allow your users to draw shapes and add text, merge in images, and drag/drop/resize graphic objects. Create and respond to HotSpots for HyperGraphic applications. Automatic Links between objects maintain connection even as independent objects are dragged around. Of course we offer full support for Scrolling, Zooming, Printing and File I/O.



**UpdateLive** is a utility you ship with your application to keeps your software and data current on an end-user system. UpdateLive can be run at the end of an install kit to check your web site for updates. UpdateLive can be run from a start menu allowing users to update anytime. UpdateLive can be run at system startup to check for updates on a daily basis. UpdateLive makes its own HTTP connection to your server, checks to see which files (if any) need updating, shows the customer what's new. UpdateLive then downloads the appropriate files, creates a backup directory (to allow reversal of the process), registers any OCX's if necessary, adds desired registry entries, and runs any Exe's if necessary.



**TList** - Never satisfied, we keep striving to improve our most popular control adding the features users want most.

Significant speed enhancements - TList is the FASTEST TREE ANYWHERE!!!

Columns/Grid support - the entire tree may be shown as a collapsable grid, or individual items may have grids as children. Multiple Bitmaps per item - one in each column  
Virtual Items - keep only those items in memory actually needed.

Unlimited named data items associated with each element of the list.

Search/Sort based on any column or hidden data item

Stand-Alone TreeDesigner application - allows building and formatting tree/grid without a line of code.

Transparent Background or Background Images - great for MultiMedia

Internet Support; 3-D look; OLE Drag&Drop; Hide Rows and Columns; LevelDef

formatting,

Much more: Parent property, Prev and NextSibling properties ...

Also:

PRINTING - very flexible powerful print engine - Headers, Footers, Zoom, Automatic page numbering, Column Headers on each page, ...

RTF formatting of each node,

Automated Grid Cell Editing

Nodes Collections Objects, Enhanced Sorting, Automated Drag/Drop,

Simple Expand To Level method

Double Byte Character Sets (japanese, chinese, ....).|

Retricted end-user views, and more.

Visual Studio .Net and Windows XP Support

Grid Cell Navigation and Selection

Enable or Disable Rows, Columns, or Cells

Grid Cell and Column References by Value Name

3-D TreeLines support

3-D Shadowed Text Support

Drag Drop Enhancements and Column Dragging

Automated Column And Row Resizing

Enhanced Format Support

Enhanced Control over Scrolling

Margin Offsets for Each Column or Cell

---

## Contact Information

Bennet-Tec Information Systems, Inc.

50 Jericho Tpk, Jericho, NY 11753

Phone (516) 997-5596

Fax (516) 997-5597

E-mail: [info@bennet-tec.com](mailto:info@bennet-tec.com)

Web: [www.bennet-tec.com](http://www.bennet-tec.com)

# Introduction

---

## Control features

TList allows you to arrange data as a multi-column list, Tree, Grid or Hierarchic Tree-Grid for optimal organization of your critical information. TList provides formatting with icons, colors and fonts that is ideal for easy end-user recognition. TList's support for in-place editing ( text, checkboxes, Combo, Calendar) provides an easy end-user interface as well as flexibility in data selection and validation. TList makes it easy to Drag & Drop, Save and reload data, Print, and offer restricted data views. Using TList, you can create Manufacturing Pick Lists, Checkbook Registers, Idea organizers, Concept development, Customer Lists, Reporting and the list goes on. The possibilities are endless.

There is a list of basic TList's features:

- **One Control Many Views.**  
Show as a List, a Tree, or a Grid. Supports combination -  
a) Multi-Column Tree within a Grid  
b) Grid's as Children of Tree items
- **FAST ! Huge Lists**  
Holds over to 2x10<sup>6</sup> items in a list.  
Add 20,000 rows/second Real Mode, (with just a 200 mhz pc )  
Or - Instant Performance Virtual Load
- **Full Featured Grid support**  
Present as Flat or hierarchic grid, individual rows can have child grids. - Hide or Show Row/Column Headers  
- Hide or Show grid Lines  
- Hidden Rows and Columns  
- Multi-Line word wrapping within cells  
- Distinct colors, fonts etc for every cell  
- User Resizable Columns  
- Specify Cell borderstyle  
- Text and images in each cell  
- Identify row/column clicked on  
- Parse Delimited Strings to/from Multi-Column Rows of data  
- Reorder Columns
- **Drag & Drop**  
Drag between controls or within TList. TList automatically scrolls the control during drag & drop.  
\* **Automated Drag and DropSupport** - no code required to drag and drop within TList or between TList controls.
- **In-Place Editing / Built-in Editors**  
Built in support for Text Editing, Checkboxes, Drop-Down Combobox, Date / Time Calendar.
- **Pictures**  
Distinct Images for each item in the list, or each grid cell, Background Pictures, Category Pictures, Tree Pictures, More

- **Fonts and Colors**  
Independent font styles, foreground and background colors for each item and cell (even RTF Formatting).  
Format by Row, Grid Cell, Column, or Hierarchic Level
- **Hidden Fields**  
Supports associated Item Data including variants, strings, OLE objects, pictures, integers etc. Store multiple named hidden data values for each item
- **Multi-Line Word Wrapping**  
Up to 32,767 characters per item. Pictures can be aligned with the top, middle or center of the text lines
- **Sorting**  
Sorts by Branch or by Hierarchic level. Sort by any column, or by hidden data. Sort options include Case Sensitivity, Alpha / Numeric sort, Parent Nodes first.
- **Searching**  
Searches visible text in any column or hidden data values. Find complete strings, substrings, or Starts-With.
- **Internet Support**  
Associate a URL with each item. Navigate from external applications, or from TList embedded within a web page
- **File I/O , Data Storage**  
Save or Load all data and formatting with just one line of code. TList data files can replace Databases for many applications. Of course it's easy to load from a database as well.
- **Design Time Support**  
List/Tree/Grid data and formatting may be set up at design time in WYSIWYG mode (no code required using TDesigner application - right click on TList on form and select Properties).
- **Selection**  
Supports simple, extended and Windows Explorer compatible multiple selections. Select with mouse or keyboard.
- **Categories**  
Assign items to distinct categories, independent of tree structure. Hide or show any category. Display category images.
- **Printing / Report Generation**  
Headers/footers, Zoom or Shrink, Column Titles on each page, Page #'s. Flexible yet Easy - just one line of code.
- **Virtual Items and Nested Virtual Items (tested with 2 billion items in a tree)**
- **VisualRoot property**  
Restrict view to Specified Tree Branch
- **ToolTips**
- **Background Bitmap and Transparent Background support**
- **Fast, flicker-free operations**

Here is a list of the features that are new for users of TList version 6.0 or earlier:

- **In-Place Editing / Built-in Editors**  
Grid cells editing, manual and automatic modes.  
Built in support for Text Editing, Checkboxes, Drop-Down Combobox, Date / Time Calendar.
- **Node/Nodes objects**  
Object-oriented access/manipulating with the tree.
- **Level based sorting**  
Sorting settings for all items belonging specified level.
- **Moveltem** – moving the items around the tree
- **AddAfter** – adding items after specified item
- **FullItemString and FullRowString**  
Exporting and importing the row of the tree or grid into and from plain text string.
- **Enhanced grid border styles**  
You can get an access via [TListGrid].BorderStyle property.
- **IntelliMouse support**

- **TriggerEvents property** – enhancement control over the Expand/Collapse event firing.
- **TitlesResize event**
- **Visual Studio .Net Support**  
Native TList for .NET Winforms component is available and it is a FREE upgrade for users purchasing a TList Subscription License with TList 8 OCX.
- **Windows Support**  
TList 8 is specifically designed and tested to support Windows XP and Windows Server 2003. TList 8 is formally supported for use under Windows 95, 98, 2000, NT 4, ME, XP and Windows Server 2003.  
Note – while there are no known problems with earlier versions of TList running in Windows XP, this is the first edition of TList specifically designed to supporting XP.
- **Grid Cell Navigation and Selection**  
TList 8 provides support for Cell-by-Cell, Row-by-Row, and Column-by-Column navigation and selection within Grids using Mouse or Keyboard.  
New selection modes provide for flexibility in Single or Multi-Cell Selection within a Grid and can be combined with Single and Multiple Selection of Tree items and settings for multiple ItemGrids in the same data set.  
See:  
**Activate** Method,  
**GridCellActivate/GridCellDeactivate, GridRowActivate/GridRowDeactivate** events  
**SelBorderStyle, SelBorderColor**  
**Activatable, Selectable, ActivationMode, KeyboardActivation, SelectionMode SelectionOptions**  
**Col and Row, ActiveCell, ActiveRow**  
**Selected, SelectedCell, SelectedRows, and SelectedColumns**  
properties for further information.
- **Enable or Disable Rows, Columns, or Cells**  
TList 8 provides control over the ability of an end-user to select or navigate through rows or cells. Individual rows and cells may be disabled for navigation and/ or selection.  
See: **Activatable, Selectable** properties for further information.
- **Grid Cell and Column References by Value Name**  
TList 8 supports referencing TListColDefs and TListGridCells by ValueName as well as by column index.
- **3-D TreeLines support**  
TList 8 supports presentation of 3-D style TreeLines.  
See: **TreeLinesColor, TreeLinesHighlightColor** and **TreeLinesShadowColor** properties for further information.
- **3-D Shadowed Text Support**  
TList 8 supports presentation of a shadowed Font3D style for text. This can be set for the entire TList control or for any column, row, or specific cell.  
See: **Font3D, FontShadowColor** and **FontShadowSelectedColor** properties for further information.
- **Drag Drop Enhancements**  
TList 8 now provides FULL support for OLE Drag Drop Previous TList editions supported TList as an OLE Drag Destination but not an OLE Drag Source.  
TList Automated DragDrop previously supported only within VB is now fully supported within all development environments – including within VC, Delphi, Access, even within HTML pages.  
See: **OleDrag** property for further information.
- **Column Dragging**  
TList 8 provides support for end-user dragging of columns with a mouse.  
See: **DragColumnsEnabled** property for further information.
- **Automated Column and Row Resizing**  
New methods to automatically resize columns and rows upon demand to fit data.  
New property to automatically resize columns upon user double click of column separator  
See: **AutoSizeColumn, AutoSizeRow** methods, and **AutoSizeOptions** property for further information.
- **Enhanced Format Support**  
TList 8 provides enhanced support for formatting data cells. TList 8 supports practically all VB formats.  
See: **Format** property for further information
- **Automatic Hierarchic Numbering of Row Headers**  
TList 8 can automatically assign hierarchic style numbering to row headers in a TListGrid  
See: **AutoFillRowTitles** property for further information.

- **Grid Row and Column Title Click Events**  
TList 8 provides events when the user clicks row or column titles in any Grid object in TList. These events are activated instead of the GridCellClick event for these grid cells.  
See: **GridRowTitleClick**, **GridColumnTitleClick**, and **GridCornerTitleClick** events for further information.
- **Enhanced Control over Scrolling**  
TList 8 provides increased control over Scrolling and Scrollbar positioning.  
See: **ScrollHRange**, **ScrollHPosition**, **ScrollVPosition**, **ScrollVRange** properties for further information.
- **TListRowDefs** object  
TList 8 provides a RowDef object for additional object oriented control over formatting of data within a Grid.
- **Margin Offsets for Each Column or Cell**  
TList 8 provides support for specifying a horizontal offset of text and images from the boundary of any column or grid cell  
See: **MarginTop**, **MarginLeft**, **MarginBottom** and **MarginRight** properties for further information.
- **MS TreeView Compatibility**  
TLTreeView object offers support for MS TreeView syntax, convert existing projects to TList in seconds then just add the features you need.

---

## New Features (Version 8.0)

New Features introduced in TList 8 include:

- **Recent Windows editions**  
TList 8 is formally designed, tested and support for use under Windows Vista and new Windows 7. TList 8 is formally supported for use under Windows 95, 98, 2000, NT 4, ME, XP, Windows Server 2003, Vista, and Windows 7 ( TList use is supported in 32 bit applications running on either 32 and 64 bit platforms ).
- **Enhanced Data Entry - Grid Navigation While Editing**  
TList supports Excel style navigation between grid cells while editing.  
User may simply move from cell to cell with arrow and tab keys while remaining in Edit Mode
- **Enhanced ToolTips support**  
TList can automatically display *tooltips*, showing the full text of an item, when the mouse cursor is moved over an item clipped by the borders of the control. Full control over the style and color of the ToolTip presentation is provided via the **ToolTipsMode**, **ToolTipsVisualStyle**, **ToolTipsForeColor**, **ToolTipsBackColor**, **ToolTipsDelay**, **ItemToolTip**, **ToolTip**, and **ToolTipText** properties, and **ShowToolTip** event.
- **Row Resizing**  
Allow end-users to resize rows by dragging on row dividers  
See: **AllowResizing** and **AutoSizeOptions** properties for further information.
- **Non-Scrolling Columns**  
TList 8 enhances support for Grid presentation by allowing the developer to specify a number of columns on the left as fixed – such that they remain stationary when scrolling TList horizontally. This support is accessed through the new property, **NonScrollableColumns**.
- **3D GridLines**  
New presentation styles - 3-D style GridLines  
Improve visibility for display over graphic or gradient background.
- **Smooth Vertical Scrolling**  
Smoothly scroll to allow display of very tall items (taller than TList window)

---

## License Registration

All Bennet-Tec software will run in demonstration mode for 30 day with full functionality. During this time you will see a demonstration message in the lower right corner of the control. After 30 days the software will time out and you will not be able to program with it. At this point, you must download another demonstration version to another machine, or purchase a license.



If your software is not properly registered, you will receive a message saying you do not have an appropriate license or that you are in demo mode.

When you install the software, whether you have downloaded the installation kit or you have installed from one of our disks, you are given the opportunity to register your software near the end of the installation process. You may register at this point, or at any time you choose. If you don't register at install time, you can do that any later time that is convenient to you by selecting the TList License Registration menu link in the Windows Start Menu.

The TList License Registration utility presents 2 tabs:

- **On-line Registration**

Use this tab if your computer is connected to the internet.

This will automatically communicate via the internet (without a web browser) with the Bennet-Tec License Registration Server to accept and verify license information from your system

You will need three pieces of information:

- 1) Your 11-character Serial Number  
(provided to you when you purchase a Bennet-Tec software product)
- 2) Machine Key  
(automatically determined by the license system and entered for you)
- 3) User Name  
(the name of the individual developer, not the company name)

- **Off-Line Registration**

If your computer is not connected to the internet, you may use the Off-Line tab.

You will need four pieces of information:

- 1) Your 11-character Serial Number  
(provided to you when you purchase a Bennet-Tec software product)
- 2) Machine Key  
(automatically determined by the license system and entered for you)
- 3) User Name (individual developer's name, not company name)  
(the name of the individual developer, not the company name)
- 4) Registration Code  
(you can get a registration code using any computer to access the web page <http://www.bennet-tec.com/register.htm>)

Select the appropriate tab and enter your information. All requested information must be entered. If all text boxes are not filled in, or have not been filled in properly, the "Register" button will not become activated. After entering the requested information, press the "Register" button. In response you will get a message box that confirms the successful entering of the registration information into the system – note this may take a few seconds as the system communicates with our license registration server.

To verify that you have your license properly installed on your development machine, start a brand new program and place your control on a form or dialog. If there is no registration message inside the control, then you are registered. At this point you may open any existing project. If you still get a registration message at this point, open and resave each individual form (not the project file) or dialog on which the control is used. Alternately, you may remove the control from your project and then add it back to the project

### **Important Notes:**

Your Serial Number is issued to you upon purchase of your license. It can also be found on your invoice and on your Bennet-Tec License Agreement.

The User Name is the name of the person actually doing the development with the control, not the company name and not the name of the individual who purchased the license. It is important to use the developer's name for the User Name so that we can recognize that name if and when we are called upon to offer support. Remember each license is valid for just one individual and can not be shared – if you must transfer the license at some point – just contact us and let us know the name of

the previous and future developers so we can update our databases. Put your User Name and your Serial Number in a safe place, as you must use the correct Serial Number and User Name each time you register.

Your Machine Key can be found inside a text box when you run the TList License Registration program. You need not worry about your Machine Key; it is already filled in for you.

The Registration Code will only concern you if you are registering off line. If this is the case, if you have access to any computer that is on line (it need not be your development machine), you may go to our registration web site--7 days a week/24 hours a day--at <http://www.bennet-tec.com/register.htm>. Click on the name of your control, fill in all the text boxes, then click the "Register Me!" button on the bottom of the page. This will generate your Registration Code. This code is valid for one machine only. You will need to generate a new registration code for each development machine you wish to register off line on.

### See Also

Registration Questions and Answers

## License Registration Questions and Answers

### **1. Who's name should I enter as User Name when I register ?**

Please specify your own full name. If you are registering the license then you should be the only individual developing projects with the control. Each license is registered for development use by one individual – this individual developer's name should be specified. Do not specify a company name, the person who purchased the control, or a project name.

### **2. Do end users of a compiled application created with a Bennet-Tec control require a License from Bennet-Tec? Are there any Royalties?**

No. End-users of a compiled EXE, or a web page built with a Bennet-Tec control do not require a license and there is NO Royalty or distribution fee . Users of other applications running in a run-time only mode ( for instance run-time only Access databases also do not require a license) User of environments such as MS Word which do not offer run-time only support DO require licenses.

### **3. How many individuals can use a single license?**

Each license is valid for just one individual developer. A developer is any individual loading a Bennet-Tec control, or loading a form or other ActiveX control built around the Bennet-Tec control, within a software design environment such as Visual Basic, Visual C, Delphi, Access ( with design mode enabled), FrontPage, etc.

### **4. Can licenses be shared?**

No, a separate license should be purchased for each individual working on a project. Licenses must be purchased for anyone opening a form, Dialog, or Class, or active X control built around a Bennet-Tec control, whether to write code, for testing, or simply to review code.

### **5. Can licenses be transferred?**

Licenses can be transferred, but only with written approval from Bennet-Tec, and only within a company.

### **6. Can I incorporate a Bennet-Tec control in my own Active X control?**

Yes - BUT - Each individual developer using an Active X control built around a Bennet-Tec control must still purchase a license from Bennet-Tec.

### **7. Do I really need a separate license for project members working on other areas of the application, if I am the only one working with code based on the Bennet-Tec control?**

YES - every individual opening your project in the software development or web design environment must have a license. This includes individuals working on other areas of code in the

project, reviewing the code within the design environment, or testing within the design environment. No license is required for testers running a compiled EXE, or for reviewing code in an ASCII text editor.

**8. Are Site Licenses available?**

Yes - contact our sales department for further information.

**9. How many computers may I install the license on?**

You may install your license on up to 3 computers at one time, So it is possible to install on your office computer, a computer at home, and a laptop. Or to install on a Win 95 machine, Win 98, and Win NT \* BUT \* these computers must all be used by the same individual. Remember each license is valid for just one individual.

**10. How can I remove a license from my computer?**

Click the UnRegister button on the License Registration Tab. This will generate a LOG file in the same directory which you should e-mail to us at [Support\\_License@Bennet-Tec.Com](mailto:Support_License@Bennet-Tec.Com) or print and fax to 1 516 997 5597

**11. Why do I see a license warning even after I registered my license?**

There are several possibilities:

- a) If an individual without a license opened your project files, possibly to review the code or to work on another part of the project, then his / her unlicensed status would have been stored in the project when that individual closed the project. Remember each individual working in development mode requires a separate license – even to review your code or to work on another part of the project. To allow code review without a license – send your code to the individual in an Ascii file.
- b) If you last opened the project while working in demonstration mode, the demonstration mode status may have been saved as part of the project. Open and resave each form using the control.
- c) If you have installed a new operating system or a new hard drive partition the Machine ID may have changed, in this case you need to re-register your license.

**12. Can I use the same License Registration code on Multiple Machines?**

No. The registration code is uniquely determined by the combination of User Name, Serial Number and Machine ID Code.

**13. How do I register my License if my computer is not connected to the Internet?**

You can use the Off-Line tab of the license registration system. You will be asked here for a Registration code. You can get the registration code using any other computer to access our web registration page:

[www.Bennet-Tec.Com/Register.htm](http://www.Bennet-Tec.Com/Register.htm)

**14. What if I can't find my Serial Number or I can't remember my User Name?**

If you can't locate your Serial Number, either contact us or contact the reseller from whom you purchased your software. If you can't remember your User Name, contact us and we will attempt to look it up for you.

**15. What if I follow the instructions, But still can't successfully register my license?**

Either attempt to reregister the license again, or contact us for assistance. You may call us (516-997-5596) or e-mail us ([Support\\_License@Bennet-Tec.com](mailto:Support_License@Bennet-Tec.com)) with your Serial Number, User Name and Machine Key, and we will then assist you in registering.

**16. What if I receive a message saying that my Bennet-Tec software has been registered by the Maximum Number of Users?**

This message means that the software license has already been registered, and that you are attempting to register the software license again, but you have entered a User Name that differs in some way from the one that was originally licensed. If you can't remember the original User Name, or if you would like to transfer your license to a different user, please contact us.

**17. Must I Register My Bennet-Tec OCX with Windows?**

Whether you install your Bennet-Tec software by downloading the installation kit from our web site (www.bennet-tec.com) or from our disk, your OCX is registered with Windows automatically for you as part of your software's installation process. Our license registration program has nothing to do with registering with Windows. The only time you may need to register your Bennet-Tec control with Windows is if you have multiple builds of one of our OCX's. In this case, you may have to register the particular OCX you would like your programs to access using Windows' Regsvr32.exe program.

**18. Must I Reregister My Bennet-Tec Software License After Downloading an Update?**

You should not have to reregister your software license after updating the software (except if you have purchased a major Upgrade – ie: a new edition of the product not just a minor edition update) . However, if you choose to install the update it in a different location, you may have to register it with Windows in order to be sure your program is accessing the correct version of your software.

**19. Must My Bennet-Tec Software License Be Registered on My End User's Machine?**

No, you need only run our license registration program on each development machine. You need not run our License Registration program on any end-user's machine; however, you must make sure as part of your installation kit that any Bennet-Tec OCX is registered ( not licensed) with Windows on your end user's machine.

**20. Where can I get more information?**

Write to us at [support\\_license@bennet-tec.com](mailto:support_license@bennet-tec.com)

Please include your TList serial number, or let us know if you are still working with the control in demonstration mode prior to purchase of a license.

---

## On-Line Help

TList has an on-line Help system that includes all of the information contained in this guide. To access Help on a TList property, highlight the item in the properties list window at design time and press F1.

---

## Distribution Notes

### Design Time Use:

As a reminder, TList licenses **may not** be distributed or shared among developers. A separate license must be purchased for each individual who will use the control within a design environment (one in which code can be written using TList, or the TList properties can be directly accessed). This includes consultants, maintenance engineers, and testers regardless of whether the individual is directly interacting with code related to TList.

Developers using TList based ActiveX controls must also purchase TList licenses even if they are not directly interacting with TList itself.

The TDesigner application (TDesign8.EXE) may not be distributed or shared with non-licensed users of TList.

**Application Distribution:** The TLIST8.OCX file itself may be freely distributed with compiled applications (.EXE). You should redistribute the appropriate custom control file TLIST8.OCX with your application and install it on the end-user's PC in the Windows\System or any other directory where DLLs can be found. In addition, the following Microsoft DLL's are required for support of TList ActiveX controls and should be included with your distribution:

<b>Files</b>	<b>Version</b>	<b>Description</b>
TLIST8.OCX	8.0.10 or newer	TList ActiveX Control
MFC42.DLL	6.00.8665.0 or newer	Support DLL (Microsoft Foundation Class DLL)
MSVCRT.DLL	6.00.8797.0 or newer	Support DLL (Microsoft Visual C++ Run-time Library DLL)
OLEAUT32.DLL	2.40.4275 or newer	Support for OLE Automation
OLEPRO32.DLL	5.0.4275 or newer	Note that OlePro32 and OleAut32 are codependent and must be of versions compatible with one another

Prior to distribution you should make sure each developer using TList has registered his/her license.

\* Bennet-Tec requests that developers notify us of the application name and contact point when distributing applications.

## **Installation Files**

The TList installation disk includes the following files:

<b>File</b>	<b>Description</b>
TLIST8.OCX	TList ActiveX custom control.
TLTVW3.OCX	TListTreeView 3.0 ActiveX custom control.
TLIST8.CHM	The help file
TLIST8.BAS	File with symbolic constant and function declarations
*.BAS, *.FRM, ETC	Various application code samples
TCONV.EXE	TList project converter
TVWCONV.EXE	MSTreeView to TListTreeView project converter
TDESIGN8.EXE	TList Designer Application

---

Note Only the licensed developers may have on their computers and use the design TDesigner™ Tree Builder utility, TDESIGN8.EXE. This file MAY NOT be redistributed or shared among developers!

---

---

## **Technical Support**

Technical support policies are as described on the Support page of the Bennet-Tec web site ([www.bennet-tec.com](http://www.bennet-tec.com))

# Features and programming techniques

---

## TList features and programming techniques

The TList control displays items in a hierarchic list ( a tree ) or grid. Each item can have subordinate items, which are visually represented by indentation levels. When an item is expanded its subordinate items are visible; when an item is collapsed its subordinate items are hidden. Items in the TList control can also display graphical elements to provide visual cues about the state of an item.

TList's Grid support provides for multi-column display. TList may be presented as a hierarchic grid – a grid where some rows are subordinate to others and may be collapsed or expanded, or as a tree of ItemGrids – where any row may be the parent of an independent grid. ItemGrids may each have their own grid formatting – with distinct column headers or even distinct numbers of columns.

Individual items, groups of items, grid cells, as well as the control as a whole, are exposed as objects exposed by TList and may be manipulated by setting properties and responding to events.

### **Features**

- Backward Compatibility
- Color Support
- Design-Time Support
- Display Features
- Expanding and Collapsing the Outline
- Hiding TList Items
- Grid Support
- Internet Interfaces
- Keyboard Interface
- Navigating the List – Choosing an Indexing Scheme
- Objects and Object Collections
- Picture, Selection and Double-Byte Characters Support
- Visual Elements and Hot Spots
- How to Add or Delete Items
- How to Boost Performance

How to Specify Default Properties  
 How to Specify and Work with Associated Hidden Data  
 How to Work with Virtual Items  
 How to Use TList Grids for Column/Table Data  
 How to Work with Databases  
 Using Virtual Functionality to Show a Database Structure  
 How to Use Tree Buffers to Manipulate the Tree  
 How to Support Drag Drop  
 How to Sort or Search a Tree  
 How to Access the Clipboard  
 How to Save and Load Lists - File I/O  
 How to Use Cell / Item Editing  
 How to Use Bookmarks  
 How to Assign Categories – TList Mark Support  
 How to Control the Display of Plus/Minus Pictures  
 How to Upgrade an Old TList 3/Pro OCX Project to Use New TList OCX  
 How to Upgrade an Old VBX Based Project to Use TList OCX in Place of the VBX  
 How To Detect the Version Number Of TList  
 How to Trap Right Mouse Clicks  
 How to Navigate a Web Site with TList  
 How to Print With TList  
 How to Use TList's RTF Support  
 How to Use the VisualRoot Property  
 How to Use LevelDefs for Sorting  
 How to Use TListNodes and TListNode Objects  
 How to Use IntelliMouse Functionality  
 How To Resize Rows And Columns  
 How To Use ToolTips  
 How to Copy from one TList to Another  
 How to Smoothly Scroll tall items  
 How to Support Excel Style Navigation While Editing

## Backward Compatibility

### Properties Which Can Be Set at Run Time Now

There are a number of properties which can be modified at in run-time now:

- **MultiSelect** property
- **DisableNoScroll** property
- **Scrollbars** property

### Obsolete Unsupported Properties

- **NoIntegralHeight** property – no longer supported
- **BackwardCompatible** property – no longer supported
- **TitlesXXX**, **ShowTitles** – no longer supported (use grid column titles instead)



- **ItemIntValue, ItemLngValue, ItemStrValue, ItemPicValue, ItemSngValue, ItemType** - actually these are still supported but are obsolete and may be discontinued in the future. It is strongly suggested that users make use of the more efficient **ItemTag** and **ItemValues** properties.
- **CoerceIndex** - again this is not really discontinued, but it has been found to be confusing to TList users and is obsolete. We strongly suggest that the **TranslateIndex** method be used instead.

### See also:

TList features and programming techniques

## Color Support

TList offers a great deal of flexibility in applying colors to list elements and the background. Text and background colors can be set for the control as a whole, item by item, column by column (in a grid), or even cell by cell (again in a grid). Distinct colors may be specified for tree lines, grid lines, selected items, and even tooltips.

The following properties may be used:

To Set	Use Property
Background Colors	<b>BackColor, SelBackColor ItemBackColor, DefItemCellBackColor GradientColorFrom, GradientColor To BackColorBkg</b>
Foreground Text Colors	<b>ForeColor SelForeColor ItemForeColor</b>
Tree Line Color	<b>TreeLinesColor</b>
Grid Line Colors	<b>GridLinesColor</b>
Cell Border Colors	<b>DefItemCellBorderColor</b>
ToolTipsColors	<b>ToolTipsBackColor ToolTipsForeColor</b>

Foreground and Background Colors are determined according to the following scheme:

First for the control as a whole

Then using properties specified by **LevelDefs.CellDef** settings

Then using Item specific properties

Then Column.**CellDef** definitions

Then specific Grid cells

For example:

```
TList1.ForeColor = SomeColor
TList1.LevelDefs(Level).CellDef.ForeColor = SomeColor
TList1.ItemForeColor(Item) = SomeColor
TList1.ItemGrid(Item).ColDefs(Column).CellDef.ForeColor = SomeColor
' Or
TList1.Grid.ColDefs(Column).CellDef.ForeColor = SomeColor
TList1.Grid.Cells(Row, Column).CellDef.ForeColor = SomeColor
```

## Color Values

Colors must be one of the following:

- A normal Windows RGB color. You can specify this with the **RGB** function, the **QBColor** function, or by using the Visual Basic color palette.
- A system default color. You can specify one of these by using one of the global constants for color listed in the Visual Basic file **CONSTANT.TXT**.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

Many TList color properties can accept the special color:

```
Const COLOR_TRANSPARENT = &H2000000
```

Also you can use the following constant to reset a color property to a default setting:

```
Const COLOR_DEFAULT = &H1
```

Neither dithered colors nor transparent colors may be applied to foreground color properties including: **ItemForeColor**, **ForeColor**, **SelfForeColor**, **TreeLinesColor**, **GridLinesColor**, **ToolTipsForeColor**, and **DefItemCellBorderColor**. These properties can accept only pure (non-dithered) colors.

#### See Also:

TList features and programming techniques

## Design-Time Support

The TDesigner application replaces the old TList Property Pages. Now TDesigner is started in response to right clicking TList on a form and selecting the Properties menu.

It is also possible to run TDesigner as a stand-alone application. This allows you to save files defining the TList property settings and tree structure you desire without writing any code. See the chapter, *Using TDesigner Application*, for details.

Files created with the TDesigner application can be loaded in any TList based application using the **LoadData** method at run-time. Applications can also use the **SaveData** method to create data files which can be read by TDesigner, or by other TList applications with the **LoadData** method.

#### See Also:

TList features and programming techniques

Using the TDesigner application

## Display Features

- **Color and Font Formatting** - each item in a TList may be assigned a foreground and background color as well as font characteristics (name, bold, italics, strikethrough, underline). In addition the **LevelDef.CellDef** and **ColDef.CellDef** properties may be used to apply defaults based on hierarchic indentation or column location (when working within a grid).
- **Selection Display** - the **InvStyle** and **InvImage** properties control how selected items are displayed when selected.
- **Focus Rectangle** - by default TList draws a rectangle around the most recently selected item in order to indicate which item has the focus. TList's **DrawFocusRect** property determines whether to display a focus rectangle. With **DrawFocusRect** set, the **InvStyle** determines the extent and presentation of the rectangle.
- **ToolTips** - TList can automatically display *tooltips*, showing the full text of an item, when the mouse cursor is moved over an item clipped by the borders of the control. Full control over the style and color of the tooltip box is provided via the **ItemToolTip**, **ToolTip** properties and **ShowTooltip** event.
- **Explorer Compatible Display Support** - TList's **ExplorerCompatible** property allows you to make TList look like Windows 95 Explorer Outline. In this mode TList will automatically supply plus/minus bitmaps, and set the color and style of tree lines. This property also controls TList behavior in response to a number of additional keystrokes like Ctrl-Right, Ctrl-Left etc., see *Keyboard Interface* section for details.

- **XOffset** - the **XOffset** property sets the left offset of TList elements (how far from the left edge of the control the item's text, picture, etc. begin).
- **Indentation Spacing** - TList allows you to specify the horizontal offset between hierarchical indentation levels. This is set in twips using the **ShiftStep** property.
- **Scrolling and Scrollbars** - TList allows the developer to control display of the scrollbars using the **ScrollBars** property. Programmatically scrolling the control may be accomplished using the **ScrollHorz**, **BottomIndex** and **TopIndex** properties. TList also provides events triggered BEFORE scrolling of the control, see **HScroll** and **VScroll** events. New in TList 8 – it is also possible to switch between item by item scrolling, and smooth pixel based scrolling for tall items.
- **MultiLine Text and WordWrapping** - TList supports the presentation of multiple line word wrapped text. Simply set the **DefMultiLine** property to True in order to default all items to wordwrapped text. Different items may be given distinctive word wrapping behavior (overriding the **DefMultiLine** property) by setting the **ItemMultiLine** property for the desired item. The width of the word wrapped line is controlled by the **WidthOfText** property. To control the vertical alignment of pictures next to multiple lines of wrapped text, set the **PicInMultiLine** property.

Example:  
TList.WidthOfText = .5 \* TList.Width  
TList.ItemMultiLine(5) = True  
TList.List(5) = a\_Very\_Long\_String\$  
TList.PicInMultiLine = 0 ' Picture at top

- **Tab Characters** - for the simplest display of columnar data, TList supports the use of a Tab character, Chr\$(9), to separate text within an item. The spacing between tab locations is specified by the **TabStopDistance** property. This may be used to present columns of data. For best results, make sure that the **TabStopDistance** is greater than length (in twips) of any string within one of the columns. In many cases improved display may be achieved by the use of TList's grid support.
- **Grids / Columns** - TList provides full support for displaying the entire tree within a collapsible grid of independently formatted and accessible columns with column and row titles. It is also possible to display grids/tables as children of specified items. For more information, refer to the section, *Using Tables/Grids/Columns*.
- **Caption** - TList supports an optional *caption* at the top of the control. Setting the **ShowCaption** property to True displays the caption. The text is controlled by the standard **Caption** property.
- **Background Image and Gradient Fill** - TList provides support for a *background image* (set with the **BackPicture** property) which may be tiled, centered, stretched, or aligned within the image boundaries as specified by the **BackPictureAlignment** property. TList also supports a *gradient background* set using the **GradientColorFrom**, **GradientColorTo**, and **GradientStyle** properties.
- **Transparent Background** - TList supports a *transparent background* style set by the **TransparentBackground** property. To update what is shown through the transparent background the **UpdateBackground** method is provided.

**See also:**

TList features and programming techniques

## Expanding and Collapsing the Outline

TList allows you to control whether the control automatically expands and collapses branches of the tree hierarchic grid in response to end-user actions. This behavior is controlled by the **AutoExpand** property. Various settings allow support for clicks on plus/minus pictures, and double clicks on the text of an item.

TList also provides full programmatic control over expanding and collapsing the tree. Setting the **Expand** property of a given item to True will expand that branch to show all immediate children of an item. Expanding an item whose parent is collapsed will first expand the parent and then expand the item itself. Setting the **ExpandEx** property to True expands a branch including all subordinates (not just immediate children). To expand the entire Tree use the special index of -1 with the **ExpandEx** property, set

```
TList.ExpandEx(-1) = True
```

By default, new items added to the tree are NOT expanded. Their children will not be immediately seen. To change this default behavior set the **ExpandNewItem** property to True.

TList triggers an **Expand** event when the list is expanded and a **Collapse** event when the list is collapsed. If Virtual Items need to be displayed when an item is expanded, TList will also trigger the **ItemQueryData** event at which time the needed data may be supplied by the application. The **ItemQueryData** event will be triggered after the **Expand** event.

The **ExpandChildren** event may be used to tell TList to recall the expand/collapse state of subordinate branches of a Tree when a higher level parent is collapsed.

---

Note TList saves the expand/collapse state of items when copying to a clipboard, copying to a *tree buffer* or saving to a TList data file.

---

TList 8 has a new **ExpandToLevel** property, which allows to expand/collapse the tree up to the specified level.

For further information refer to descriptions of the **AutoExpand**, **Expand**, **ExpandEx**, **ExpandChildren**, **ExpandToLevel**, and **ExpandNewItem** properties and the **Expand** event.

[See also:](#)

TList features and programming techniques

## Hiding TList Items

The new **ItemAlwaysHidden** property keeps items hidden (not visibly shown within the tree) even if its parent is expanded and other children are shown to the user. For example, if we have the following actual tree structure:

```
Item1
  Item 1.1
  Item 1.2 '(Expanded)
    Item 1.2.1
    Item 1.2.2 '(Hidden)
    Item 1.2.3
  Item 1.3 '(Collapsed)
    Item 1.3.1
    Item 1.3.2
```

The user will see the following representation on the screen:

```
Item1
  Item 1.1
  Item 1.2
    Item 1.2.1
    Item 1.2.3 ' (There is no Item 1.2.2 shown)
  Item 1.3
```

The **ItemAlwaysHidden** property is different from the TList **IsItemVisible** (read-only) property. If **ItemAlwaysHidden** is set to True, it means that this item cannot be shown until this property is set back to False. If **IsItemVisible** returns False for an item it can mean either that the **ItemAlwaysHidden** property is True, OR simply that one of this item's parents is collapsed. It is also possible to hide a group of items all at once. Use **MarkedItemsAlwaysHidden** property to make *always hidden* all items with a given mark.

## TList features and programming techniques

TList provides support for 2 types of Grids:

- |    | Accessory Store                       | Order                               | Price | Comments   |
|----|---------------------------------------|-------------------------------------|-------|--|
| 34 | Intel PIII 733 Copernine FC-PGA OEM   | <input type="checkbox"/>            | 149\$ |  |
| 35 | Intel PIII 750 Copernine FC-PGA BOX   | <input type="checkbox"/>            | 153\$ |  |
| 36 | Intel PIII 800 Copernine FC-PGA OEM   | <input checked="" type="checkbox"/> | 174\$ |  |
| 37 | Intel PIII 866EB Copernine FC-PGA OEM | <input type="checkbox"/>            | 193\$ | Column titles                                    |
| 38 | Intel PIII 933EB Copernine FC-PGA OEM | <input type="checkbox"/>            | 228\$ |  |
| 39 | 3 RAM                                 | <input checked="" type="checkbox"/> |       |  |
| 81 |                                       |                                     |       |  |
| 82 |                                       |                                     | 41\$  |  |
| 83 | - DIMM 128MB PC-133                   | <input type="checkbox"/>            | 42\$  |  |
| 84 | - DIMM 128MB                          | <input type="checkbox"/>            | 46\$  |  |
| 85 | - DIMM 128MB                          | <input checked="" type="checkbox"/> | 45\$  | Regular grid cells with checkboxes and text data |
| 86 | - DIMM 128MB PC-133 SEL               | <input type="checkbox"/>            | 50\$  |  |
| 87 | - DIMM 256MB PC-133                   | <input type="checkbox"/>            | 78\$  |  |
| 88 | - DIMM 256MB PC-133 Hyundai           | <input checked="" type="checkbox"/> | 89\$  |  |
| 89 | - DIMM 256MB PC-133 Micron            | <input type="checkbox"/>            | 85\$  |  |
| 90 | - DIMM 32MB PC-100                    | <input type="checkbox"/>            | 24\$  |  |
| 91 |                                       | <input type="checkbox"/>            | 25\$  |  |
| 92 |                                       | <input type="checkbox"/>            |       |  |
| 93 | Video cards                           | <input type="checkbox"/>            |       |  |
|    | Hard                                  | <input type="checkbox"/>            |       |  |

- 
- The screenshot shows a software interface with a tree view on the left and a table on the right. The tree view has three main categories: 'Removable Drives', 'CD-ROM drives', and 'Sound Cards'. The 'Removable Drives' category is expanded, showing three items: '1.44MB Mitsumi', '100MB Zip int', and '250MB Zip int'. The 'CD-ROM drives' category is also expanded, showing one item: 'C-Media 8738 4-channel'. The 'Sound Cards' category is expanded, showing a list of items including 'Creative SB Live! 5.1 Player', 'Creative SB Live! Platinum 5.1', 'Creative SB128PCI', 'Creative SB128PCI + SB535 KIT', 'Crystal 4235 3D', 'Diamond Sonic Impact S100', 'FM Tuner PCI', 'Genius SM Live 4.1', and 'Media-FAST-5000 Quad M...'. The table on the right has four columns: 'Model', 'Delivery', 'Guarantee', and 'Comments'. The table contains data for the items listed in the tree view. Annotations highlight the 'Parent item that owns ItemGrid object' and 'Children of the parent item which form correspondent ItemGrid object'.
- | Model          | Delivery  | Guarantee | Comments              |
|----------------|-----------|-----------|-----------------------|
| 1.44MB Mitsumi | 2-3 days  | Months    | No comments           |
| 100MB Zip int  | 2-3 days  | ee months | Call if out of stock  |
| 250MB Zip int  | 5-7 days  | o months  | Call before processor |
|                | Overnight |           |                       |
- | Model                          | Delivery  | Guarantee  | Comments              |
|--------------------------------|-----------|------------|-----------------------|
| C-Media 8738 4-channel         | Overnight | Two months | No comm               |
| Creative SB Live! 5.1 Player   | 2-3 days  | None       | Call before           |
| Creative SB Live! Platinum 5.1 | 2-3 days  | Two months | No comm               |
| Creative SB128PCI              | 2-3 days  | None       | Call if out           |
| Creative SB128PCI + SB535 KIT  | 2-3 days  | 6 months   | Call if out           |
| Crystal 4235 3D                | vs        | None       | No comments           |
| Diamond Sonic Impact S100      | h         | 6 months   | Call before processor |
| FM Tuner PCI                   |           | 6 months   | Call before processor |
| Genius SM Live 4.1             |           | Two months | No comments           |
| Media-FAST-5000 Quad M...      | ad        | None       | Call before stock     |



It is also possible to mix the two types of grids – TList may be formatted as a TreeGrid where certain rows of the grid have their own grids as children.

Both grids use the same set of properties and methods. **TListGrid** object is responsible for either type of grid.

Visually a grid has the following parts:

- Column titles - shown in row 0 of a TreeGrid or ItemGrid. Use the **TList1.Grid.Cells(0, XXX).Text** property to set/retrieve the column title string.(xxx refers to the specific column).
- Columns - provides an interface for specifying default formatting for all cells belonging to a particular column. The **TList1.Grid.ColDefs(ColIndex).CellDef** property may be used to change these settings.
- Row titles – optionally shown as the first column in a TreeGrid. Use the **Grid.Cells(XXX, 0).Value** property to set/retrieve row title strings. Use **TList1.Grid.ShowRowTitles** to hide or show the column holding row titles. Note that RowTitles are shown only for TreeGrids, not for ItemGrids.
- Rows - each item provides data for one row. The **RowHeight** property may be used to set the height of any row.
- Grid cells - each item has **Values** collection of **Value** objects. Each value object provides data for a cell of an item's row. Use **TList1.Grid.Cells(Row, Col).Value** to set/retrieve cell values.  
Regular grid cells can hold a data(text) and a picture. Grid cell with tree item in it allows you to manipulate with grid as if it is a tree (collapse/expand branches, create a hierarchy etc).

#### See also:

How to Use TList Grids for Column/Table Data  
TList features and programming techniques

## Internet Interfaces

TList 8 has several methods to facilitate Web navigation: **WebAutoNavigate**, **WebGoBack**, **WebGoForward**. These methods work regardless of the container TList is loaded in. For example, if TList is placed on a Visual Basic form and the **WebNavigate** method is executed, the default Web browser installed on the computer will be started and the specified Web document will be located and shown to the user. (Currently only Internet Explorer v 3 and above are supported).

You can make TList navigate through the Web automatically when the user double clicks on a list item displayed in the control. TList uses the **ItemURL** and **TListCellDef.Url** properties to specify the URL for a hop. **WebTargetFrame**, **WebURLBase**, and **WebAutoNavigate** properties control operation of this mode. These properties are saved as part of TLT data files, which can then be referenced on a Web page or loaded into an application using the **SaveData** method.

You can use VBScript <Object> tag attributes to resize TList to fit its container on an HTML page.

#### See also:

TList features and programming techniques

## Keyboard Interface

End-Users may use the keyboard to select items in a TList control's list. The following table lists the

keys and their actions:

This key	Moves focus
UP ARROW	To the previous item, if any
DOWN ARROW	To the next item, if any
HOME	To the first item that is visible
END	To the last item that is visible
PAGE UP	Backward one page, or to the first item currently displayed
PAGEDOWN	Forward one page, or to the last item currently displayed

In addition, you can use two keys to scroll horizontally in a list, this is useful when the length of an item's text is larger than will fit within the width of the control.

Key	Action
RIGHT ARROW	Scroll to the right
LEFT ARROW	Scroll to the left

If the **ExplorerCompatible** property is set to "1 - Keystrokes" or "3 - Keystrokes and Tree Lines appearance", the following keystrokes are also processed:

Key	Action
SHIFT-LEFT OR LEFT	Moves <b>ListIndex</b> to the parent of current parent or closes the current parent.
SHIFT-RIGHT OR RIGHT	Expands current parent or moves <b>ListIndex</b> to the first child of the current parent.
CTRL-RIGHT	Scrolls control to the right (if the horizontal scrollbar is visible) without changing of the <b>ListIndex</b> .
CTRL-LEFT	scrolls control to the left (if the horizontal scrollbar is visible) without changing the <b>ListIndex</b> .
CTRL-HOME	Scrolls control to the top without changing the <b>ListIndex</b> .
CTRL-END	Scrolls control to the bottom without changing the <b>ListIndex</b> .
CTRL-UP	Scrolls control one item up without changing the <b>ListIndex</b> .
CTRL-DOWN	Scrolls control one item down without changing the <b>ListIndex</b> .
CTRL-PAGEUP	Scrolls control one page up without changing the <b>ListIndex</b> .
CTRL-PAGEDOWN	Scrolls control one page down without changing the <b>ListIndex</b> .
PLUS	Expands currently selected item.
MINUS	Collapses currently selected item.
ASTERISK	Expands/collapses all items

**See also:**

TList features and programming techniques

## Navigating the List - Choosing an Indexing Scheme

Manipulation of an outline requires a mechanism for identifying individual elements within the control. TList actually supports three distinct methods of enumerating items in the list. The method in current use is set by the **CurrentIndexMethod** property.

The three methods are:

1. Enumeration of all items counting from the first (0) to the last. This is the default setting.
2. Enumeration including only visible items.
3. Enumeration by path name plus index to immediately subordinate items.

These methods are reviewed below. The method in use at any time is specified by the **CurrentIndexMethod** property. The **TranslateIndex** method may be used to translate an index from one index method to another.

1. **Enumeration of ALL items:** With the **CurrentIndexMethod** property of the control set to its default value of 0 (also specified by the constant TLSYS\_ENUM) each item in the control is enumerated with an index according to its position from the top of the list. The index of the first item is 0. The index of the last item is equivalent to the number of items minus 1 (TList.ListCount - 1).

TList control	Accessing	Notes
Item1	<u>Index 0</u>	
Item1.1	<u>Index 1</u>	
Item1.2	<u>Index 2</u>	◀ Current item
Item1.2.1	<u>Index 3</u>	invisible items
Item1.2.1.1	<u>Index 4</u>	
Item1.2.1.2	<u>Index 5</u>	
Item1.2.2	<u>Index 6</u>	
Item1.3	<u>Index 7</u>	
Item1.3.1	<u>Index 8</u>	
Item2	<u>Index 9</u>	

2. **Enumeration of Visible Items:** Setting the **CurrentIndexMethod** property to a value of 1, (specified by the constant TLSYS\_VIS) indexes items in a manner similar to the first method but enumerating only visible items.(items which may be seen by scrolling within the list - without expanding any parents not already expanded).

TList control	Accessing	Notes
Item1	<u>Index 0</u>	
Item1.1	<u>Index 1</u>	
Item1.2	<u>Index 2</u>	◀ Current item
Item1.2.1	Can not be accesed	invisible items
Item1.2.1.1	Can not be accesed	
Item1.2.1.2	Can not be accesed	
Item1.2.2	Can not be accesed	
Item1.3	<u>Index 3</u>	
Item1.3.1	<u>Index 4</u>	
Item2	<u>Index 5</u>	

3. **Enumeration by Path Name and Index:** With the **CurrentIndexMethod** property set to 2, TList is navigated by reference to a path (similar to a DOS directory path) pointing to a parent item, and only the subset list of items which are direct children of the specified parent is enumerated. The parent of the sublist is referred to as the *current parent* and the path to this item is specified using the **CurrentParent** property.



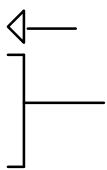
---

Note setting the **CurrentParent** property does NOT change the **ListIndex** property and does NOT change the selection of items within the Tree.

---

The *path* specifies the location of an item within the tree. Think of it as the route TList must travel, starting at the root item, to get to items that are subordinate to another item. In specifying the path, each item except for the root item (which is always represented by the string as defined in the **PathSeparator** property) has a name. The name is the same as the item's displayable text, which is specified by elements of the **List** property array.

For example, in the tree pictured below, "Item 1.2 " is the current parent and only "items 1.2.1" and "Item 1.2.2" would be enumerated - they would be list elements 0 and 1 respectively:

TList control	Accessing	Notes
Item1	Can not be accesed	 <p>Current item</p> <p>Visibility not considered in indexing scheme.</p>
Item1.1	Can not be accesed	
Item1.2	Can not be accesed	
Item1.2.1	<u>Index 0</u>	
Item1.2.1.1	Can not be accesed	
Item1.2.1.2	Can not be accesed	
Item1.2.2	<u>Index 1</u>	
Item1.3	Cannot be accesed	
Item1.3.1	Cannot be accesed	
Item2	Cannot be accesed	

To specify the sublist parent, set the **CurrentParent** property by reference to its path. Each item in the control has a name. The name is the same as the item's displayable text, which is specified by elements of the **List** property array. A path specification may consist of the item names separated by a path delimiter character (as set by **PathSeparator** property). Alternatively an item in the path may be specified by its position within its parent's list of subordinates.

For example to specify Item 1.2 in the list above set:

TList1.CurrentParent = "\\Item1\\Item1.2"

or

' indicating child #1 of Root #0  
 TList1.CurrentParent = "\\#0\\#1"

or

' indicating child #1 of Root item named "Item1"  
 TList1.CurrentParent = "\\Item1\\#1"

or

' indicating child named "Item 1.2" of Root #0  
 TList1.CurrentParent = "\\#0\\Item 1.2"

All items (both visible and invisible) immediately subordinate to this current parent are then indexed sequentially.

Only items subordinate to the current parent can be accessed by index in this way. Neither the current parent itself, nor any item not subordinate to the current parent can be accessed by index. To access such items using this indexing method, you must designate a new current parent. To access item(s) of the upper level (such as Item 1 and Item 2 in the outline above), you must change the current parent to the root item.

Actually the current parent may be referred to by Index of -1 when in this index mode, or by index -2 at all other times.

The path of any subordinate item may be determined using the **FullPath** property. It is then possible to change the current parent from 1.2 to 1.2.2 in the list above using the statements:

```
x$ = TList1.FullPath(1) 'Returns the full path of element indexed as 1
TList1.CurrentParent = x$
```

The **CurrentParent** property can accept either string or long specifying the index of a new parent.

#### [See also:](#)

### **Special Index Values**

Regardless of the setting of the **CurrentIndexMethod** property settings, TList has 3 special *index values*, which may be used to quickly identify key items in the list.

- -1 refers to the parent of the list. With the **CurrentIndexMethod** property set to 0 or 1, this means the TList control as a whole. Use this index with the **ExpandEx** property to expand/collapse the entire tree, or with **Copy...** and **Add** properties to copy an entire list or to add to the end of the current list. With the **CurrentIndexMethod** property set to 2, the parent of the list is the same as the item specified by the **CurrentParent** property.
- -2 refers to the item specified by the **CurrentParent** property. This is most useful when the **CurrentIndexMethod** property is set to 2.
- -3 refers to the most recently added item, or the item whose hierarchic indentation has most recently been changed. Using an index of -3 is faster than reading the **NewIndex** property.

#### [See also:](#)

Navigating the List – Choosing an Indexing Scheme TList features and programming techniques

### **Finding an Item's Parent**

TList provides an **ItemParent** property, which may be used to find the immediate parent.

---

Note **ItemParent** will always return -2 when used in conjunction with **CurrentIndexMethod** property set to 2.

---

The **FullPath** property may also be readily parsed to identify the complete ancestry of a given item.

#### [See also:](#)

Navigating the List – Choosing an Indexing Scheme

### **Finding the Next Sibling**

To get the next sibling of an item read the **ItemNextSibling** property.

---

Note this will always return the next child of the same parent. Likewise the **ItemPrevSibling** property will return the previous sibling of the same item. A value of -4096 indicates no previous or next sibling.

---

#### [See also:](#)

Navigating the List – Choosing an Indexing Scheme

### **Finding the Last Subordinate Item for the Specified Item**

To get the index of the last subordinate item for a specified item read the **ItemLastSubItemIndex** property.

#### [See also:](#)

Navigating the List – Choosing an Indexing Scheme

## Finding the Parent

It is easy with TList to get the parent item for any item in the list. Note that the ListIndex property points to the item with the focus.

Here are two examples of how to get the Parent of that item

- 1) To get the Index of this Item's parent read the ItemParent property

```
With TList ' substitute name you have assigned to TList
  ItemIndex = .ListIndex
  ParentIndex = .ItemParent( ItemIndex )
  ParentsText = .List ( ParentIndex )
End With
```

- 2) Using TListNode objects

```
Dim TListNode as TListNode
Dim TLParent as TListNode
With TList
  NodeIndex = .ListIndex
  TListNode = TList1.Node ( NodeIndex ).Parent.text
  TLParent = TListNode.Parent
  ParentText = TLParent.Text
End With
' the same as above but written in one line
ParentText = TList1.Node ( TList1.ListIndex ).Parent.text
```

## See also:

TList features and programming techniques

Navigating the List – Choosing an Indexing Scheme

## Objects and Object Collections

Starting with version 4, TList now exposes several new interface elements as a set of objects providing access to a variety of new design elements.

TList objects include:

- **TListGrid** - refers to a grid object (can be a TreeGrid holding the entire TList, or an ItemGrid holding just a grid of items subordinate to some parent).
- **TListValue** - holds an associated data element. Normally not visible, but it can be assigned to a Grid column for display.
- **TListLevelDef** - manages default formatting for items of a given hierarchic indentation level.
- **TListColDef** - describes a Grid column including column titles
- **TListCellDef** - holds a standard set of attributes like background color, text color, font etc.
- **TListGridCell** - describes a Grid cell
- **TListReport, TListPage** - manages reporting/printing functionality, supports complex printing beyond that offered by PrintOneStep method.
- **TListNodes, TListNode** - object-oriented access to the items in the tree.
- **TListEditInfo, TListCheckbox, TListCombobox** etc - manages built-in editing support with textboxes, checkboxes, comboboxes, date/time and spin controls.
- **TListRowDefs, TListRowDef** - row oriented control over formatting of data within a Grid.
- **TListSelectedGridRows** - a collection of TListRowDef objects that are currently selected.

- **TListSelectedGridCells** - a collection of TListGridCell objects representing all selected cells in the grid.
- **TListSelectedGridColumn**s - a collection of TListColDef objects that are currently selected.
- **TListTooltip** - an object, the properties of which specify the content and presentation of the tooltip window to be displayed
- **TListShowTooltipArgs** - an object that defines the content and presentation of a tooltip. This object is presented as a parameter of the ShowTooltip event, giving the programmer full control over tooltip appearance and behavior.
- **TListTooltipStyle** - an object that defines the colors and text alignment used for display of a tooltip window.
- **TListTooltipWindow** - an object that defines the location and size of a tooltip window.

As you see, all TList object names begin with the prefix "TList".

TList also supports a number of object collections (**TListLevelDefs**, **TListColDefs**, **TListValues**, **TListPages**, **TListNodes** etc). All object collections have a **Count** property and an **Items** property. These properties may be used to enumerate all existing objects stored in an object collection. Note that objects can be referenced in a number of ways, either as members of an objects collection, or by a TList property returning an object. These Object Reference properties include:

- **ActiveGrid** - refers to a TListGrid object
- **Grid** - refers to a TListGrid object
- **ItemGrid** - refers to a TListGrid object
- **LevelDefs** - refers to a TListLevelDef object
- **ItemCell** - refers to a TListCellDef object
- **Values** - refers to a TListValue object
- **Cells** - refers to a TListGridCell object
- **Node** - refers to a TListGridCell object
- **RowDef** - refers to a TListRowDef object

The example below illustrates how Visual Basic's *For Each* statement can be used to enumerate all data associated with item 100 (this code assigns the same value 777 to all data for the item):

```
Dim TListValues1 As TListValues
' TList1.ItemValues(100) returns a TListValues object
TListValues1 = TList1.ItemValues(100)
Dim X As TListValue
For Each X In TListValues1
    X = 777
Next
```

The sample rewritten in shorter form:

```
Dim X As TListValue
For Each X In TList1.ItemValues(100)
    X = 777
Next
```

The next example sets alternating background colors for each column of a grid:

```
Dim TListColDefs1 As TListColDefs
TListColDefs1 = TList1.Grid.ColDefs
Dim X As TListColDef
For Each X In TListColDefs1
    X.BackColor = QBColor(x.Index mod 16)
```

```
Next
```

Or in shorter form:

```
Dim X As TListColDef
For Each X In TList1.Grid.ColDefs
    X.BackColor = QBColor(X.Index mod 16)
Next
```

or:

```
For I = 0 To TList1.Grid.ColDefs.Count
    TList1.Grid.ColDefs(I).BackColor = QBColor(I mod 16)
Next
```

Finally here is an example showing the setting of alternating colors based on the hierarchic level of an item in a tree:

```
Dim I as Integer
For I = 0 To 255 ' for each possible indentation level
    'Set alternating default back and fore colors
    TList1.LevelDefs( I ).CellDef.BackColor _
        = QBColor( I Mod 16 )
    TList1.LevelDefs( I ).CellDef.ForeColor _
        = QBColor( 15 - I Mod 16 )
Next
```

### See also:

TList features and programming techniques

## Picture, Selection and Double-Byte Characters Support

TList provides support for the display of many images within a list. The following **Picture** and **Image** properties are available:

- Plus/Minus pictures - set with **PicturePlus** and **PictureMinus** properties. Display of these images is determined by the **ViewStyleEx** property.
- Mark Pictures - displays images based on an item's *item mark* category - set by assigning images to the **MarkPicture** array, and setting the **ItemMark** property for a given item to reference the array. Display of these images is also determined by the **ViewStyleEx** property. Mark pictures are displayed along the left margin of the control window.
- Item image (Open, Closed, Leaf, Root) - the **PictureLeaf**, **PictureClosed**, **PictureOpen**, and **PictureRoot** properties are used to determine the image based on whether an item has children and the item's expand/collapse state. Note that these picture properties may be set for TList as a whole or to a **TListLevelDef** object to define defaults based on *heirarchic indentation level*. To display the main image the **ViewStyle** property must be less than or equal to 3. Also the **PictureType** property should be set to its default value of 0.
- Item Image (individual Image) - specifies a distinct individual image for individual TList items, set the **Image** property for that item. This overrides **PictureOpen**, **PictureClosed**, etc, on an item-by-item basis. The **InvImage** property may also be set to specify a distinct image for when the item is selected. Again, the **ViewStyle** property must be less or equal to zero in order for the item image to be displayed.
- Multiple Cell Pictures - every item in a TList has one or more cells (multiple cells are displayed if the item is also a row of a **TreeGrid** or **ItemGrid** with multiple columns). Each cell can contain both text and a picture. The picture is assigned with the **Picture** property of a **CellDef** object. Setting the **PictureSelected** property tells TList to display a different image when the item is selected. The alignment of the picture in a cell with respect to text is controlled by the

**Alignment, DefItemCellAlignment, DefItemCellPictureAlignment, and CellPictureAlignment** properties.

```
TList1.ItemCell(1).Picture = Picture1.Picture  
TList1.Grid.Cells(2,2).CellDef.Picture = Picture1.Picture  
TList1.ItemCell(1).PictureSelected = Picture1.Picture
```

As with Fonts and Colors, Images may be defined on an Item by Item basis, by Level, by column or by cell.

See also: **PicInMultiLine** property; **PlusMinusClick** ; **PlusMinusDbfClick**; **PictureClick**; **PictureDbfClick**.

You can reset the images referenced by the various picture properties using 2 methods:

1. Using the **Nothing** Visual Basic constant.

```
Set TList1.Grid.GridCellDef.Picture = _  
    LoadPicture("NotEmpty.BMP")  
Set TList1.Grid.Cells(5, 5).CellDef.Picture = _  
    Nothing
```

2. Using the **LoadPicture** function result.

```
Set TList1.Grid.GridCellDef.Picture = _  
    LoadPicture("NotEmpty.BMP")  
Set TList1.Grid.Cells(5, 5).CellDef.Picture = _  
    LoadPicture()
```

The first method (setting the picture to **Nothing**) truly resets the picture property as if it had never been set. In the second method, **TList** assumes that there is a real (but empty) picture is specified for an object.

The difference is important only for the **Picture** and **PictureSelected** properties. If there is a default picture defined (for instance a default cell picture may be set by with the **ColDef**, **CellDef**, **Picture** property, or a default may be set with **LevelDef**, **GridCellDef** objects) then this default image will be displayed after removing the item's picture property with the first method. The default image will not be displayed if the item image is cleared with the second method.

## Palette Support

To force all pictures to be displayed with the same color palette use **PalettePicture** property.

## Transparent Bitmap Support

**TList** supports a transparent color for bitmaps. To set this mode on, use **TransparentBitmap** property. To specify the transparent color use **TransparentBitmapColor** property.

---

Note Setting of **TransparentBitmap** property to **True** will result in slightly slower refreshing of **TList** display.

---

## Selection Support

**TList**'s **MultiSelect** property allows the developer to specify whether a user may select one item at a time, or multiple items. Selections may be made by mouse click, by dragging, or by programmatic setting of the **Selected** and **SelectEx** properties.

The **SelItemCount** property returns the total number of selected items.

The **SelItemIndex** property contains an array of all the indices of selected items. Reading this property returns the index of the N<sup>th</sup> selected item.

The **Selected** property is an array of all elements in the list which are selected. To select an item programmatically set the **Selected** property for that item to **True**:

```
TList.Selected(5) = True
```

To determine if an item is selected, read the **Selected** property.

To select all subordinate elements of a given item, set the **SelectEx** property. Specifying an index of -1 selects the entire list (as defined by the **CurrentIndexMethod** property).

The **CopySelected** property copies all selected items to a *tree buffer* from which they may be saved to a file, or inserted to another location of the same or a different tree.

---

Note By default the changes to the **SellItemIndex**, **SellItemCount**, and **ListIndex** properties all occur BEFORE the **MouseDown** event is triggered. In some cases (particularly when conducting drag drop operations in a multi-select list, it may be useful to have TList modify these properties only after the **MouseDown** event is completed. The **SmartDragDrop** property may be set for this purpose.

---

See the **InvStyle**, **InvImage**, **SelectedPicture**, **SelForeColor**, **SelBackColor** properties for information on controlling the display of selected items.

### Double-Byte Characters Support

TList accepts full Double Byte characters within any string properties.

Most TList string oriented properties accept any ASCII values except for ASC(0) which is usually treated as a termination character. The only properties which accept ASC(0) as a valid character within a string are **ItemTag** and **ItemValues**.

#### See also:

TList features and programming techniques

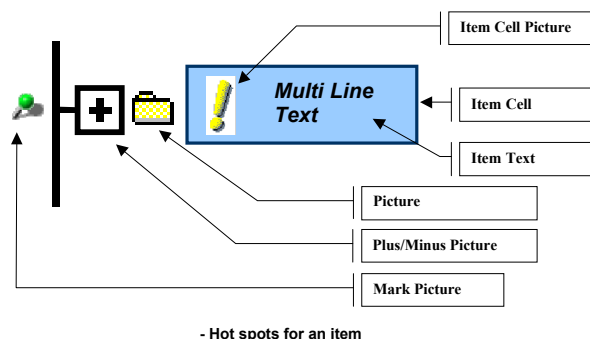
## Visual Elements and Hot Spots

TList is organized as a hierarchic list. Each member of the list is generally referred to as an *item*.

The hierarchic indentation level is specified by the **Shift** property and the items are indexed/enumerated in order as specified by the **CurrentIndexMethod** property.

TList can display graphics and text for each item in a list. An item can have several visual elements. The **ViewStyle** and **ViewStyleEx** properties control which of these elements are displayed.

Each graphical element - plus/minus, tree lines, text, and pictures - is a *hot spot* graphic. Clicking a hot spot triggers a special set of events. In addition if the **WebAutoNavigate** property is set, double clicking on an item will activate the Web navigation features of TList. The following diagram shows an item's possible hot spots.



- **Plus/Minus Picture** - a picture on which the user can click in order to expand or collapse the tree. The picture is set using the **PicturePlus** and **PictureMinus** properties. Clicking on the bitmap will trigger **PlusMinusClick** and **PlusMinusDblClick** events. Additionally the tree may be collapsed or expanded depending on the **AutoExpand** property. Setting the **ExplorerCompatible** property automatically sets the images for the plus and minus bitmaps.
- **Tree Lines** - vertical and horizontal lines that link items with subordinate items. Tree lines may be drawn in a variety of styles, solid, dashed, and are controlled by the

**TreeLineStyle** property. The color of the tree lines may be set with the **TreeLinesColor** property. Setting the **ExplorerCompatible** property automatically sets the **TreeLineStyle** and Color to emulate Windows 95 Explorer

- **Picture** - the picture displayed for an item. You can specify the default images, which are displayed depending on the state of the item using the **PictureRoot**, **PictureOpen**, **PictureClosed**, **PictureLeaf**, and **PictureInverted** properties. Specific images may be set for each item in the list (overriding the **Picture** defaults) by setting the **Image** and **InvImage** properties. The **PictureType** and **NoPictureRoot** properties indicate how TList identifies which images to display. Clicking on the picture will result in **PictureClick** and **PictureDbClick** events.
- **Item Cell** - the region in which the text is displayed for an item. Item cells may have their own distinct back color from the rest of the item. In fact in the case where data is presented in grid/columnar fashion, there will be several item cells - one for each column - each of which may have its own foreground color, background color, gradient fill, and its own *item cell picture* and *text*.
- **Item Cell Picture** - an image displayed with the text in the *item cell* portion of an item. The alignment of the image with respect to the text is controlled by **DefItemCellAlignment** and **Alignment** properties, while the alignment of the picture within its area of the cell is controlled by the **DefItemCellPictureAlignment** and **PictureAlignment** properties. Use the **PictureWidth** and the **PictureHeight** properties to control the size of the displayed item cell picture.
- **Item Text** - text displayed for an item. Initially set when using the **AddItem** or **InsertItem** methods, the text may also be read and manipulated by reference to the **List** property (an indexed array). Clicking on the text will generate click and or double click events. The font, foreground and background colors may be specified for each item using **ItemFont...**, **ItemForeColor** and **ItemBackColor** properties. Alignment of the text within the text area of an item cell is controlled by the **DefItemCellTextAlignment** and **TextAlignment** properties. Within a grid, formatting for each column in an item may be independently set for each item cell. TList 8 offers accepts and displays RTF formatted text for any item or grid cell text,. Use the **RTFStyle** property to control the way TList displays RTF formatted text.
- **Indentation** - an item's level of subordination. Each level of indentation is a level of subordination specified by the **Shift** or **Indent** property. The **ShiftStep** property determines the horizontal offset between hierarchic levels.
- **Mark Picture** - a category based picture displayed for an item, selected from an array of *mark pictures*. TList supports the assigning of a *mark* (a numeric category identifier) to each item in the outline. Each mark may have a picture assigned to it (using the **MarkPicture** array). Setting the **ViewStyleEx** property to 2 or 3 causes TList to display the mark picture along the left margin of the control window for any elements having an associated mark with a specified mark picture. Clicking on the mark picture triggers **MarkClick** and **MarkDbClick** events.

#### See also:

TList features and programming techniques

## How to Add or Delete Items

A full set of operations for reorganization of a structure of the list is available with TList control. The following table summarizes the basic methods of adding and deleting elements from the list.



How to:	You can use:
Add an item to the end of the list.	The <b>AddItem</b> method.
Add an item as a new last child of an item.	The <b>AddItem</b> method with second parameter (or, using the <b>CurrentIndexMethod</b> property set to 2, set the <b>CurrentParent</b> property to point to the desired parent, and use <b>AddItem</b> method without second parameter).
Insert an item before another item at the same indentation level.	The <b>InsertItem</b> property, (or the <b>AddItem</b> method with second parameter and with the <b>MSOutlineAdd</b> property set to True – functions just like MSOutline control).
Remove an item.	The <b>RemoveItem</b> method.
Remove all subordinate items from an item.	The <b>ClearItem</b> property.
Remove all items from a list.	The <b>Clear</b> method.

The behavior of the **AddItem** method is controlled by the **MSOutlineAdd** property. This property provides MSOutline control compatibility.

Other mechanisms for adding items to a list include:

- Building the list using the TDesigner application
- Loading data from a TList data file (see section on File I/O)
- Loading data from a *tree buffer* (frequently used in Drag Drop operations)
- Pasting items from the clipboard
- Adding virtual items to a TList

The simplest way to specify *columnar* type data is to use delimiters when adding items to TList. Just set the **ConvertTabsToCols** and **ColDelimiter** properties before calling the **AddItem** method.

```
TList1.ConvertTabsToCols = True
TList1.ColDelimiter = Asc("^")
TList1.AddItem "Fred^Jones^SomeStreet^New York"
TList1.AddItem "Sarah^Lenore^Another Ave^Florida"
```

TList provides a wide range of properties and objects to allow specification of every aspect of a grid. Refer to sections on *Using TList Grids* and *Using TList with a Database* for further information.

### See also:

TList features and programming techniques

## How to Boost Performance

Bennet-Tec has listened carefully to the needs of TList users over the years and recognizes the importance of performance. We have totally restructured TList internally to maximize performance especially when building a tree. In addition the following notes will help you to achieve the best performance possible based on your application needs:

Always set the **Redraw** property to False before making extensive changes to the list, and reset it back to True when you are done making changes. This will greatly enhance performance by eliminating the need for many screen updates.

---

**Important** Use the special index value "-3" to reference the most recent item added (or whose indentation was changed) in TList. This is significantly faster than using the **NewIndex** property.

---

When setting individual images for items in TList, remember that Visual Basic's **LoadPicture** function always returns a reference to a unique image even if loading the same image many times. It is MUCH faster and much more efficient in system resources to load the image just once and then reference it repeatedly. See the **Image** property description for more details

To build a list with many item specific formatting details, use the **AddItem2** and **AddItem2Ex** methods, not the **AddItem** method. These new methods are the easiest way to add items one by one to the end of the tree while setting the item formatting (font, font style, and color) at the same time. If you have a huge tree with many items which may or may not be seen by the end user (depending on the expand state of the tree) you may also want to consider using Virtual items which are only loaded when TList needs to refer to them. Setting the **ItemVirtualCount** property tells TList that a given item has a certain number of virtual children. When the end-user expands the list, TList will generate an **ItemQueryData** event in which you can then supply it with the necessary data. This will save memory and cut down on the time to build and display the initial tree

For really impressive performance use TList's builtin File/I/O capabilities to *Save* a complete tree to a file and load the tree from the file as needed. Loading a tree from a TList data file is significantly faster than building the tree one item at a time. If a tree will be modified by an end-user, you can make use of this feature by saving the tree before exiting the application and reloading next time the application starts. Even if the tree is based on an external data source you can save the TList data file each time and simply compare the dates on the external data source and the TList data file to determine if the Tree really needs to be rebuilt.

#### See also:

TList features and programming techniques

## How to Specify Default Properties

TList allows the user to set up multiple levels of inheritance to determine the fonts, colors and images to be used to display an item, or even an individual cell of an item.

Fonts and colors (foreground and background) are determined according to the following scheme:

1. First for the control as a whole:

```
TList1.ForeColor = SomeColor  
' or  
TList1.FontName = SomeFontName
```

2. Then using properties specified by **LevelDef.CellDef** object settings:

```
TList1.LevelDefs(Level).CellDef.ForeColor = SomeColor
```

3. Then using item specific properties:

```
TList1.ItemForeColor(Item) = SomeColor
```

4. Then **ColDef.CellDef** object definitions:

```
TList1.ItemGrid(Item).ColDefs(Column).CellDef.ForeColor = SomeColor  
' Or  
TList1.Grid.ColDefs(Column).CellDef.ForeColor = SomeColor
```

5. Then specific grid cells:

```
TList1.Grid.Cells(Row, Column).CellDef.ForeColor
```

#### See also:

TList features and programming techniques

## How to Specify and Work with Associated Hidden Data

TList offers a number of ways to store associated data with each TList item (row): these data elements can be recalled for a given item, used as search or sort criteria (see *How to Sort or Search a Tree*), or displayed as data in a grid (see *How to Use TList Grids for Column/Table Data*).

**ItemTag Property** - The simplest way to store associated data is to use the **ItemTag** property. This array property can store one data element per item. The data can be String, Integer, Long, Single, **Picture**, even OLE or Variant.

**ItemXXXValue Properties** - Users of past editions of TList may also recall the *ItemIntValue*, *ItemStrValue*, *ItemLngValue*, *ItemSngValue* and *ItemPicValue* properties. These are still supported for compatibility purposes but they are limited, inefficient and obsolete. Future support is not guaranteed and so these properties should be avoided.

**ItemMark and MarkTag Properties** - The **ItemMark** property holds an integer value from 0 to 255. While not really intended as a data storage mechanism it is a very useful way of categorizing items. Setting an **ItemMark** value associates a TList Item with an element from the **MarkTag** and **MarkPicture** array. With **ViewStyleEx** set to 2 or 3, The appropriate mark picture will be shown for each item. Moreover, whole groups of items can be hidden or displayed quickly by using the **MarkedItemsAlwaysHidden** property

**Values Objects** - TList also provides a mechanism for storing an unlimited number of named associated data (*values*) for each TList item (*row*). These values are accessible via the **Values** property, which returns a reference to an object collection in which all values are stored. To access a value you can use a statement such as:

```
TList1.ItemValues(100, "FirstName").Value = "John"
```

The "FirstName" string is a *name*, which is used as a key, to find data you are interested in. There is no such operation as add-a-new-value. A new value object is created whenever you ask for its data for the first time.

```
TList1.ItemValues(100, "SecondName").Value = "Smith"  
TList1.ItemValues(100, "Age").Value = 100
```

To check whether a value really exists use the **ItemHasValue** property (this avoids accidentally creating an object value by trying to read a value, which may not yet exist):

```
if TList1.ItemHasValue(100, "Age") Then  
MsgBox "Age is specified!"  
End If
```

The **ItemHasValue** property may also be used to remove a value from the item:

```
TList1.ItemHasValue(100, "SecondName") = False  
or  
TList1.ItemValue(100, "SecondName") = Nothing
```

To remove all values from the item use the following statement:

```
TList1.ItemHasValue(100) = False  
or  
TList1.ItemValues(100) = Nothing
```

If you need to add many values to the same item, you can optimize performance by referring to the **Values** collection:

```
Dim ValuesOfTheItem As TListValues  
Set ValuesOfTheItem = TList1.ItemValues(100)  
ValuesOfTheItem("SecondName") = "Smith"  
ValuesOfTheItem("CityName") = "Los Angeles"  
ValuesOfTheItem("Age") = 100  
ValuesOfTheItem("Salary") = 3456.456
```

In addition, TList allows text and picture data to be stored in the **MarkTag** and **MarkPicture** arrays and associated with given items by category using the **ItemMark** property.

Note that named data elements may be readily displayed in a TList grid. See the section *Using TList Grids* for further information.

**See also:**

TList features and programming techniques

## How to Work with Virtual Items

**ItemVirtualParent** and **ItemVirtualCount** are powerful properties that eliminate the need for building the complete tree list at once, resulting in significant performance gains and memory/resource savings when dealing with very large lists.

Instead of immediately adding each item by calling the **AddItem** method, only items to be always kept in memory are added directly (for instance using the **AddItem** method). Place holders for the remaining items (children of directly added items) are added by setting the **ItemVirtualCount** property to indicate the number of virtual children for a given parent item. The control then triggers an **ItemQueryData** event to notify the application when data is needed (for example when a parent having virtual children is expanded), so the application can supply the data to the control.

The **ItemVirtualCount** property is used internally by TList to calculate the minimum and maximum values for the control's vertical scroll bar, the delta for each of the scrollbars thumb positions, and the proper index values for items later in the list.

The Item for which **ItemVirtualParent** or **ItemVirtualCount** property was set is called "virtual parent". These items have their **ItemVirtualParent** property set to True and the **ItemVirtualCount** property  $\geq 0$ . Children of virtual parents are called "virtual children".

When the control needs data, such as when an item with virtual children is expanded or when a property of a Virtual child is read, the **ItemQueryData** event is fired.

```
Sub TList1_ItemQueryData (ByVal ItemIndex As Long, _  
    ByVal SiblingIndex As Long)
```

Inside the **ItemQueryData** event subroutine, the application should fill in the data requested by the control. The ItemIndex parameter indicates which item, out of the entire list in the tree, needs to be loaded with data. The SiblingIndex indicates which virtual child out of all the parent's children is being loaded. That's all there is to it. TList makes virtual data trees simple.

---

Note Each item added using the **Add** property can have virtual children. But an item can have either only virtual children or non-virtual ones, not both.

---

Virtual Child Items cannot have additional non-virtual children, but they can have virtual children and any number of virtual grand children:

```
Private Sub TList1_ItemQueryData(_  
    ByVal ItemIndex As Long, ByVal SiblingIndex As Long)  
    TList1.AddItem "Child1",ItemIndex  
    TList1.AddItem "Child2",ItemIndex  
End Sub
```

Virtual children are not kept in memory forever: they are automatically discarded when not needed and when the control has time to remove them. When the control next needs to use data of discarded items it fires the **ItemQueryData** event again. Only virtual items which have subordinates or whose **ItemVirtualParent** property is set to True are kept in memory forever.

It is possible to add children to virtual children. It is also possible to make virtual children virtual parents:

```
TList1.Clear  
TList1.AddItem "Non-Virtual Item"  
TList1.ItemVirtualCount(0) = 100  
TList1.ItemVirtualCount(1) = 1000
```

```
TList1.ItemVirtualCount(50) = 10000  
TList1.AddItem "Non-virtual kid of a virtual parent", 10
```

This feature is not limited by the nesting level.

**ItemQueryData** is not fired for virtual children which have children or for which **ItemVirtualParent** property was set to True. Such items are called "fixed".

**ItemQueryData** is fired only when absolutely necessary, so it will not be generated, for example, for items which are not visible. It is also generated only once for visible items and not generate again if TList's window needs repainting; only scrolling or property changes will fire this event again.

**ItemQueryData** is fired when virtual child property settings are retrieved:

```
TList1.Clear  
TList1.AddItem "Non-Virtual Item"  
TList1.ItemVirtualCount(0) = 100  
...  
MsgBox TList1.List(50)
```

In this sample **ItemQueryData**(50, 48) is triggered when the **List** property is read.

Selection is kept separately from the virtual items so it is possible to preserve multiple selections for virtual children (even if there are millions of them).

To make virtual items of zero indentation, use -1 index with the **ItemVirtualParent** and **ItemVirtualCount** properties:

```
TList1.ItemVirtualCount(-1) = 1000
```

---

Note All previously existing TList items are removed after this property call.

---

### Virtual Children Limitations

Virtual Children cannot be hidden or sorted. When copied or saved to a file only virtual children with subordinates are really saved.

A Virtual parent can have only virtual children, it is not possible to apply **AddItem** or **InsertItem** operations on it. But virtual children of such parents can have either virtual or non-virtual children (not both together).

Bookmarks can be retrieved for Virtual children, but they are not valid as soon as the item for which bookmark is asked for is unloaded, this is very short time and therefore we can consider that bookmarks cannot be used with Virtual items.

Virtual Children can be edited (using **ItemEditText** property) but the data will be lost when the item is removed from memory.

### See also:

TList features and programming techniques

## How to Use TList Grids for Column/Table Data

There are 2 types of grids which can be shown in TList:

- **Tree Grid** - this wraps the entire TList tree within a collapsable grid as shown below:





Creating a Grid  
 Referencing Grid Cells  
 Cell Formatting  
 Determining Active Grid Cells and Selection  
 How Row Numbers and Item Indexes Compare  
 Using ValueNames and ItemValues to Set Column Data  
 How to Specify Different Settings For Even and Odd Rows In a Grid  
 TList features and programming techniques

## Creating a Grid

To create a Grid or to modify the number of columns, set the **Grid.Cols** property. For example:

```

TList1.Grid.Cols = 6 ' Specifies a TreeGrid with 6 columns
' Specifies 5 columns in the Grid held by item 1
TList1.ItemGrid(1).Cols = 5
or
' add a new column to the TreeGrid
TList1.Grid.Cols = TList1.Grid.Cols + 1
  
```

The default characteristics of cells in the table can then be set with the **ColDefs** object

```

TList1.Grid.ColDefs(1).Width = 1440 ' width of column 1 = 1 inch
or
TList1.ItemGrid(1).ColDefs(3).BackColor = QBColor(2)
  
```

To set the number of rows use the **Grid.Rows** property.

Use the **AddRow** method to quickly add several values to a row:

```

' This code adds a new row as a last row in the Grid.
TList.Grid.AddRow _
    "RowTitle" & Chr$(9) & "TextOfCol 1" Chr$(9) & "TextOfCol 2"
  
```

Tab characters are used in the **AddRow** method as a delimiter to identify which data goes in which column. The string preceding the first column delimiter is considered a Row title and will not be shown if the **Grid.ShowRowTitles** property is false. Setting the **ConvertTabsToCols** property will cause TList to similarly parse data presented with the **AddItem** method. The column delimiter may be changed with the **ColDelimiter** property.

## See also:

How to Use TList Grids for Column/Table Data

## Grid cell/row/column navigation and selection

TList provides new type of navigation for the grid objects – cell by cell navigation in addition to the standard row-by-row navigation (controlled by **ActivationMode** property of TListGrid object).

This new type of navigation will co-exist and co-interact with current types of single and multiple selection (controlled by **MultiSelect** property of TListGrid or main TList objects). So user will be able to enter to the grid, navigate inside it using keyboard or mouse and select/deselect multiple cells or just a single one, select/deselect rows and columns depending on a status of **MultiSelect** property. The **KeyboardActivation** property controls the end-user's ability to navigate through the tree/grid structure using the keyboard (enable/disable using TAB, ARROWS etc keys).

**Selectable** property of the **TListCellDef** object allows user to enable/disable selection over the cells/rows/columns. **Activatable** property of the **TListCellDef** object allows user to enable/disable the navigation over the cells/rows/columns.

The programming access to the selection interface is available via **SelectedCells**, **SelectedColumns** and **SelectedRows** properties of the **TListGrid** object. **ActiveCell**, **ActiveRow**, **Col** and **Row** properties allows user to set/get what cell/row is active.

Here is a source code that shows using new navigation and selection functionality:

```

TList1.Grid.SelectionMode = TL_SELMODE_MULTIPLE
TList1.Grid.ActivationMode = TL_ACTIVMODE_CELL

Shape1.BackColor = RGB(255, 255, 200)
Shape2.BackColor = RGB(255, 255, 150)
Shape3.BackColor = RGB(255, 255, 50)

'Disable activation of all cells in column 2
TList1.Grid.ColDefs(2).CellDef.Activable = TL_ACTIV_DISABLED

'Disable selection of all cells in column 4
TList1.Grid.ColDefs(4).CellDef.Selectable = TL_SEL_DISABLED

'Disable activation of all cells in row 10
TList1.Grid.RowCellDef(10).Activatable = TL_ACTIV_DISABLED

'Disable selection of all cells in row 12
TList1.Grid.RowCellDef(12).Selectable = TL_SEL_DISABLED

'Disable activation of cell(3,3)
TList1.Grid.Cells(3, 3).CellDef.Activable = TL_ACTIV_DISABLED

'Disable selection of cell(5,1)
TList1.Grid.Cells(5, 1).CellDef.Selectable = TL_SEL_DISABLED

TList1.Grid.ColDefs(3).CellDef.Format = "**Generic* month"

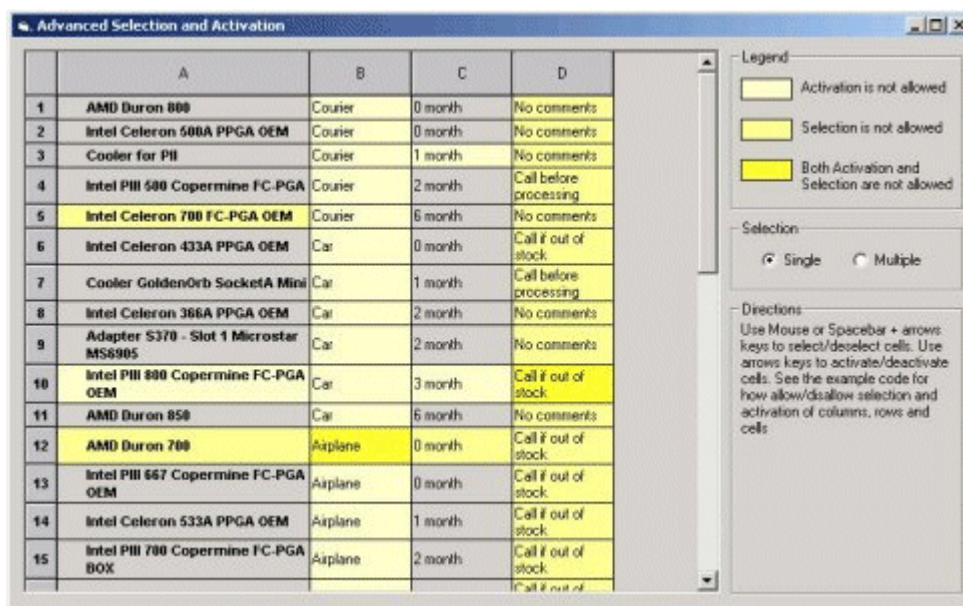
'Allow Select columns and rows by their captions click
TList1.Grid.SelectionOptions = TL_SELOPT_SELECTCOLS_ONTITLECLICK Or
TL_SELOPT_SELECTGRID_ONCORNERCLICK Or TL_SELOPT_SELECTROWS_ONTITLECLICK

'Highlight Cells that we do are not allow select and/or activate
Dim I as Long, j as Long
For i = 1 To TList1.Grid.Rows - 1
    For j = 1 To TList1.Grid.Cols - 1
        If (TList1.Grid.Cells(i, j).CellDef.Activable = TL_ACTIV_DISABLED And _
            TList1.Grid.Cells(i, j).CellDef.Selectable = TL_SEL_DISABLED) Then
            TList1.Grid.Cells(i, j).CellDef.BackColor = RGB(255, 255, 50)
        End If
        If (TList1.Grid.Cells(i, j).CellDef.Activable <> TL_ACTIV_DISABLED And _
            TList1.Grid.Cells(i, j).CellDef.Selectable = TL_SEL_DISABLED) Then
            TList1.Grid.Cells(i, j).CellDef.BackColor = RGB(255, 255, 150)
        End If
        If (TList1.Grid.Cells(i, j).CellDef.Activable = TL_ACTIV_DISABLED And _
            TList1.Grid.Cells(i, j).CellDef.Selectable <> TL_SEL_DISABLED) Then
            TList1.Grid.Cells(i, j).CellDef.BackColor = RGB(255, 255, 200)
        End If
    Next j
Next i

```

Here is an appearance of the sample created using source code above:  
 (note that the real data structure was created/loaded in design-time using TDesigner application)





### See also:

Activate Method,  
GridCellActivate/GridCellDeactivate, GridRowActivate/GridRowDeactivate events  
SelBorderStyle, SelBorderColor, Activatable, Selectable, ActivationMode, KeyboardActivation, SelectionMode,  
SelectionOptions, Col and Row, ActiveCell, ActiveRow  
Selected, SelectedCell, SelectedRows, and SelectedColumns properties for further information.

## Display of Row and Column Headings

The display of Column and Row titles is determined by the **Grid** (or **ItemGrid**) **ShowRowTitles** and **ShowColTitles** properties. By default TList will enumerate column titles "A, B, C, D, .... Z, AA, AB, ....", and Row titles "1, 2, 3, 4". **Grid.AutoFillRowTitles** and **Grid.AutoFillColTitles** properties control this display.

### See also:

How to Use TList Grids for Column/Table Data

## Referencing Grid Cells

There can be only one TreeGrid in a TList object. It is referenced as TList.**Grid**.

ItemGrids are owned by individual items in a list and are referenced by the standard TList index number as in TList1.**ItemGrid**(1).

Grid cells belonging to either a TreeGrid or ItemGrid may be referenced by the **Cells** property's row and column numbers (starting with row 0 for column titles row and column 0 for row titles column).

### See also:

How to Use TList Grids for Column/Table Data

## Accessing Cell Data and Pictures

The Text contained in the various cells is determined by the **Values** property of each cell.

```
TList1.Grid.Cells(0,1).Value = "Column Heading 1"
TList1.Grid.Cells(5,0).Value = "Row Heading 5"
TList1.Grid.Cells(3,3).Value = _
    "This text will be shown in Row 3, Column 3"
TList1.ItemGrid(3).Cells(2,2).Value = _
    "show me in row 2, column 2 of grid owed by item 3."
```

### See also:

## How to Use TList Grids for Column/Table Data

### Cell Formatting

Individual cells may be formatted within TList using the CellDefs object. For example:

```
TList1.Grid.Cells(2,2).CellDef.Font.Name = "Arial"  
TList1.Grid.Cells(2,2).CellDef.ForeColor = QBColor(1)
```

#### [See also:](#)

How to Use TList Grids for Column/Table Data

### Determining Active Grid Cells and Selection

Clicking in a TList triggers a **GridCellClick** event. It is possible to determine where he clicked by reading the **Grid**, **Row** and **Col** properties of the referenced GridCell object. Moreover the **ActiveGrid** property will be reset by TList to that **Grid**.

The **MouseRow** and **MouseCol** properties of a TList **Grid** determine where the mouse is during a **MouseMove** or **MouseDown** event.

---

Note neither **ItemClick** nor **ItemDbClick** events are fired for items belonging to a Grid.

---

#### [See also:](#)

How to Use TList Grids for Column/Table Data

### How Row Numbers and Item Indexes Compare

In addition to referencing Grid cells by row/column coordinates, each row in a grid also corresponds to a distinct indexed TList item. Row 0, which contains column titles, is an exception - this row is not considered a separate item but is part of the item which parents the grid. Thus row numbers and standard TList list indexes will NOT coincide. Enumerating items in TList, indices start with 0 and increment for every row - not including column titles. Row counts start with the column title row in a grid as row 0.

Assume the following grid setup

```
TList1.Clear  
TList1.Grid.Cols = 5  
TList1.AddRow "Row1" & Chr$(9) & "Fred" Chr$(9) & "Smith"  
TList1.AddRow "Row2" & Chr$(9) & "Paul" Chr$(9) & "Jones"
```

In this example, Column titles (row 0) have not to be explicitly set (they will be automatically assigned by TList). The first row added with the **AddRow** method is row 1. This corresponds to the first indexed item (index =0). The second row added here (row 2) corresponds to the 2<sup>nd</sup> TList item (index 1). If we now read the **List** property we will see that both **TList.Grid.Cells(1,0).Value** and **TList.List(0)** return "Row1" while both **TList.Grid.Cells(2,0)** and **TList1.List(1)** return a value of "Row2".

The rest of the data has been parsed into columns and may be retrieved with either **Grid.Cells(row,col)** property or with **ItemValues** property. Thus both **TList1.Grid.Cells(2,1).Value** and **TList1.ItemValues(1, "I1").Value** return a value of "Paul". ("I1" is the default value name, which automatically generated for a new added column 1, "I2" would be the default value name for column 2)

---

Note The TListGrid objects have an **ItemIndexToRow** method which can be used to translate from index to row Number

---

#### [See also:](#)

How to Use TList Grids for Column/Table Data

## Using ValueNames and ItemValues to Set Column Data

TListView objects have an **ItemIndex** property which returns the index of the item which owns the value:

```
MsgBox "Index = "& TList1.Grid.Cells(20,10).Value.ItemIndex
```

Using ValueNames give us flexibility in grouping of data. Using this approach we can easily specify what data are to be displayed in a column. As explained above, each row in a TList Grid corresponds to a standard TList item. The **ItemValues** data corresponds to data in the cells. Now remember that **ItemValues** may be referenced by number or by **ValueName**. The **ValueName** property may also be assigned to a **ColDef** object instructing TList to take associated data elements using that name and display the values in the appropriate column.

```
TList1.Grid.ColDefs(1).ValueName = "FirstName"  
TList1.Grid.ColDefs(2).ValueName = "SecondName"  
TList1.Grid.ColDefs(3).ValueName = "CityName"
```

Now, if an item has a value which has a **ValueName** of "FirstName" , this data will be displayed in the 1<sup>st</sup> column:

```
TList1.ItemValues(1, "FirstName").Value = "Ann"  
TList1.ItemValues(1, " SecondName ").Value = "Smith"  
TList1.ItemValues(1, " CityName ").Value = "Los Angeles"  
  
TList1.ItemValues(2, "FirstName").Value = "Jane"  
TList1.ItemValues(2, " SecondName ").Value = "Smith"  
TList1.ItemValues(2, " CityName ").Value = "Atlanta"  
  
' 1 | 2 | 3  
' Ann | Smith | Los Angeles  
' Jane | Smith | Atlanta
```

Changing the **ValueName** associated with a Column will immediately update the control and show the desired data in the column, unless the **Redraw** property is set to False. Likewise changing the **ItemValues** will update the displayed data for **ColDef**'s pointing to that data.

[See also:](#)

How to Use TList Grids for Column/Table Data

## How to Specify Different Settings For Even and Odd Rows In a Grid

Use the RowCellDef property to specify default settings for the whole row, for example:

```
TList1.Grid.RowCellDef(9).BackColor = RGB(0, 0, 0)  
TList1.Grid.RowCellDef(10).BackColor = RGB(127, 127, 127)
```

This will make TList to display 9<sup>th</sup> and 10<sup>th</sup> rows of the tree grid in different background colors.

[See also:](#)

How to Use TList Grids for Column/Table Data

## How to Work with Databases

TList is NOT a bound control.

There are, however, many ways of showing a database structure with TList.

[See also:](#)

## Navigating Records of a Table

Lets assume a database such as might be used for a Bill of Materials for some manufactured equipment, or for a company's organizational chart. Thus we have a database table where records refer to parts (or organizational groups), each of which may also have sub-parts. There is nothing

special distinguishing parts and sub parts and they may be mixed in the same table. The level of hierarchy is essentially an ancestral tree. So we want to display this hierarchy in TList.

- Assume a database table with the following fields:

```
' DisplayText - text to be shown in the list
' ParentText - text shown in parent, blank if a root item
' ItemType - a numeric category, perhaps used to indicate fabricated or purchased part
' AssociatedData1 - perhaps the price of the part
' AssociatedData2 - perhaps the Vendor Name
```

- Open the database:

```
Dim DB As Database
Set DB = OpenDatabase(x)
```

- Open a Recordset, sorting the records by ParentText, this will insure when building the list that the parents are loaded first:

```
Dim RS as RecordSet
Set RS = TABLENAME ' NEED TO SPECIFY SOME TABLE HERE...
```

- Loop through all records, adding items to their parents:

```
TList1.ReDraw = False ' don't update display while building the list
RS.MoveFirst
Do Until RS.EOF      ParentName = RS.Fields("ParentText").Value      ParentID =
TList1.FindItem(ParentName,0,0,TList1.ListCount-1)  DisplayText = RS.Fields("DisplayText").Value      TList1.AddItem
DisplayText, ParentID  Index = TList1.NewIndex      TList1.ItemMark(Index) = RS.Fields("ItemType").Value
      TList1.ItemValues(Index, "Price") = RS.Fields("Price").Value TList1.ItemValues(Index, "Vendor") =
RS.Fields("Vendor").Value      RS.MoveNextLoop
```

- Display price and vendor information in columns (visible data already in column 1):

```
TList1.Grid.ColDefs(2).ValueName = "Price"
TList1.Grid.ColDefs(3).ValueName = "Vendor"
```

- Visibly update the list:

```
TList1.ReDraw = True
```

### See also:

Using Virtual Functionality to Show a Database Structure  
TList features and programming techniques

### Using Virtual Functionality to Show a Database Structure

What follows is an example of how TList can be used to view all non-system tables of any database, using the Virtual Children features of TList to minimize the data actually held in memory at any time:

- Assume a database of the following structure:

```
Table1
  FieldAttr1      FieldAttr2 FieldAttr3
  Field1   Field2   Field3
  Field1   Field2   Field3
Table2
  FieldAttr1      FieldAttr2 FieldAttr3
  Field1   Field2   Field3
  Field1   Field2   Field3
Table3
  FieldAttr1      FieldAttr2 FieldAttr3
  Field1   Field2   Field3
  Field1   Field2   Field3
```

- Open the database:

```
Dim DB As Database
Set DB = OpenDatabase(x)
```

- Create root level (zero indentation) TList items which refer to Table Names:

```
Dim I As Long
For I = 0 To DB.TableDefs.Count - 1
    If (Left$(DB.TableDefs(I).Name, 4) <> "MSys") Then
        TList1.AddItem DB.TableDefs(I).Name
    End If
Next I
```

- Fill in Grid objects of root level items, with column headers information:

```
Dim J As Long
TList1.ItemGrid(-3).Cols = DB.TableDefs(I).Fields.Count + 1
For J = 1 To DB.TableDefs(I).Fields.Count
    TList1.ItemGrid(TList1.NewIndex).Cells(0, J).Value = _
        DB.TableDefs(I).Fields(J).Name
Next J
```

- Set **ItemVirtualParent** property:

```
Dim RS As Recordset
Set RS = DB.OpenRecordset(DB.TableDefs(I).Name)
If Not RS Is Nothing Then
    If Not RS.EOF And Not RS.BOF Then
        TList1.ItemVirtualParent(TList1.NewIndex) = True
    Else
        TList1.ItemVirtualParent(TList1.NewIndex) = False
    End If
End If
```

- We are done with primary filling of TList:

```
End If
Next I
```

- Now TList tree looks like:

```
Table1
Table2
Table3
```

- When the user double clicks on a node, the node expands and **Expand** event is generated:

```
Private Sub TList1_Expand(Index As Long)
    'We need to make sure that the number of virtual items in
    ' this branch match the number of records in the database

    Dim RS As Recordset
    If TList1.ItemVirtualParent(Index) = True Then
        If TList1.ItemVirtualParent(Index) = True Then
            Set RS = DB.OpenRecordset(Node.Text, _
                dbOpenDynaset)
            RS.MoveLast
            TList1.ItemVirtualCount(Index) =
                RS.RecordCount
            TList1.ItemTag(Index) = RS
        End If
    End If
End Sub
```

- Now, when TList needs to draw an item, it starts generating **ItemQueryData** events to fill in table's fields, there you need to fill in Node object with proper information:

```
Private Sub TList1_ItemQueryData(ByVal Index As Long, _
    ByVal, ByVal SiblingIndex As Long)
    Dim I As Long
    Dim ParentIndex As Long
    ParentIndex = TList1.ItemParent(Index)
    Dim Grid As TListGrid
    Set Grid = TList1.ItemGrid(ParentIndex)
```

```
Dim RS As Recordset
Set RS = TList1.ItemTag(ParentIndex)
If Index <= RS.RecordCount Then
    RS.AbsolutePosition = Index
    Dim GridCell As TListGridCell
    For I = 0 To Grid.Cols - 1
        Set GridCell = Grid.Cells(0,I)
        If Not IsNull(RS.Fields(GridCell.Text)) Then
            TList.ItemValues(Index,GridCell.Value.ValueName)_
                = RS.Fields(GridCell.Text)
        End If
    Next I
End If
End Sub
```

[See also:](#)  
How to Work with Databases

## How to Use Tree Buffers to Manipulate the Tree

TList allows the developer to copy items from the outline to a buffer, the *tree buffer*. The *tree buffer* is simply a memory area pointed to by a long integer, created so you can easily edit the tree structure. This buffer stores information about one or more items in the list. A new *tree buffer* is created whenever any of the properties **CopyItem**, **CopyOne**, **CopySelected** or **CopyItemSub** are read (returning a long integer pointing to the *tree buffer*). You can then copy those items to another location by writing the *tree buffer* value (as returned by the copy properties) to either the **Insert** or **Add** property. Once created, a *tree buffer* value is valid within your application for any instance of the TList control so you can even copy elements of a tree between TList controls.

There may be up to 10000 buffers in the system simultaneously (if we have enough memory for storing items which are in these buffers). When a *tree buffer* is no longer needed, the associated memory should be released by calling the **FreeBuffer** method. After that, the buffer value is no longer valid (it doesn't point to anything), and any property that accepts such a *tree buffer* value will generate trappable error. It is possible to check if a *tree buffer* value is valid using the **IsValidBuffer** method.

How to	You can use
Copy an item with its subordinate items to a <i>tree buffer</i> .	<b>CopyItem</b> property
Copy an item without its subordinate items to a <i>tree buffer</i> .	<b>CopyOne</b> property
Copy an item's subordinate items to a <i>tree buffer</i> .	<b>CopyItemSub</b> property
Copy selected items with their subordinate items to a <i>tree buffer</i> .	<b>CopySelected</b> property
Insert item(s) from a <i>tree buffer</i> before a specified item.	<b>Insert</b> property
Add item(s) from a <i>tree buffer</i> to the end of the subordinate item(s) list of a specified item (as children of the item).	<b>Add</b> property

---

### Notes

1. Adding new items to TList **does not** automatically expand the new item's parent (if any) to show the newly added item unless the **ExpandNewItem** property is set to True. This will depend on the **Expand** property setting corresponding to the parent item, or the setting of the **ExpandNewItem** property.
2. When you no longer need the *tree buffer*, free the associated memory by calling the **FreeBuffer** method.
3. Each new item added to the list adds approximately 16 bytes plus the length of the text to the memory requirements of TList. Individually assigned fonts, colors and images add to the memory usage.
4. To prevent the control from updating while items are being removed or inserted use the **Redraw** property as shown below:

```
TList1.Redraw = False
' Some operations
' maybe some AddItem or ItemForeColor settings.
' End of operations
TList1.Redraw = True
```

5. Bookmarks are NOT preserved for items copied to a *tree buffer*.

---

### See also:

TList features and programming techniques

## How to Support DragDrop

The following list identifies features of the TList control you can use to implement drag drop within your application:

- Automated DragDrop handling mechanisms - see DragDrop: Automated Support Technique
- Standard Manual Drag Drop Mechanisms: **DragMode** and **DragIcon** properties and **DragOver**, and **DragDrop** events. - see DragDrop: Manual DragDrop Technique
- Support of TList as an OleDragDrop Target - see DragDrop:- OLE DragDrop Technique
- A **DragHighlight** property determines the appearance of items as other items or controls are dragged over them.
- A **DropTarget** property provides for easy identification of the item under the mouse while dragging.
- *Tree buffers* and clipboard support to facilitate copying items from one control to another(see *How to add, delete, or copy item(s)* ).
- Automatic Scrolling during Drag Drop. This is controlled using the **AutoScrDuringDragDrop** property. We could have made the property name longer but our fingers got tired.
- Smart selection of a currently dragged item. This is controlled using the **SmartDragDrop** property. If you set this property to True, items won't change their selection state while they are being dragged (as would otherwise happen when you depressed the mouse to initiate dragging).

---

**Note** for C++ users of TList 3/Pro VBX - The OCX editions of TList don't support (or require) **DragEx**, **DragIconEx** properties or the **DragOverEx** and **DragDropEx** events as required in the VBX edition.

---

## Drag & Drop Requirements



To use TList Drag Drop features within the OCX edition of TList you must include the **OnDragOver**, **OnDragDrop** and **BeforeDrag** methods. Failure to call these methods will result in an error condition. These methods should be called as follows:

- Call the **OnDragOver** method in the first line in the **DragOver** event:

```
Private Sub TList1_DragOver(Source As Control, X As Single, _
    Y As Single, State As Integer)
    TList1.OnDragOver X, Y, State
    ...
End Sub
```

- Call the **OnDragDrop** method in the first line in the **DragDrop** event:

```
Private Sub TList1_DragDrop(Source As Control, X As Single, _
    Y As Single)
    TList1.OnDragDrop X, Y
    ...
End Sub
```

- Call the **BeforeDrag** method before the call to **Drag** method, which initiates the Drag Drop:

```
TList1.BeforeDrag
TList1.Drag 1
```

### See also:

Automated DragDrop Support  
 Manual DragDrop Support  
 How to Support OLE Drag Drop  
 TList features and programming techniques

### **Automated DragDrop Support**

*(Note: this feature is formally supported only for Visual Basic programmers.)*

TList's Automatic DragDrop feature minimizes the effort required to implement drag/drop functionality in an application. In this mode TList automatically handles initiation of Drag/Drop, setting of the Drag Icon, and copying or moving the dragged item to its destination. Note that it is possible to drag/drop items between several controls inside one application. AutomaticDragRequest and AutoDragComplete events also provide for simple customization of the drag drop process.

1. Call the **OnDragDrop** method as the first statement in the **DragDrop** event procedure  
 Call the **OnDragOver** method as the first statement the **DragOver** event procedure  
 These calls are REQUIRED for any TList control which is going to participate in drag/drop.  
 example:

```
Private Sub TList1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
    TList1.OnDragOver X, Y, State
    ...
End Sub

Private Sub TList1_DragDrop(Source As Control, X As Single, Y As Single)
    TList1.OnDragDrop X, Y
    ...
End Sub
```

2. Specify the desired **AutoDragMode** setting:  
 This determines whether a dropped items is placed before, after, or as a child of the item it is dropped upon.

```
TList1.AutoDragMode = TLAUTODRAGMODE_ASCHILD 'Drop item(s) as child
```



That's it. TList will automatically handle the rest. By default TList will MOVE items being dragged, or copy them if the end-user depresses the Cntrl key while dropping the item.

TList's **AutoDragRequest** and **AutoDragComplete** events allow further customization of automatic drag drop operations. The **AutoDragRequest** event may be used to preventing starting of a drag/drop operation for any item. The **AutoDragComplete** event may be used to control the completion of a drag/drop operation so you can easily cancel the drag/drop operation if, for example, certain items are not allowable drag drop targets, or to control whether an item is moved or copied as a result of the drag/drop operation. :

```
Private Sub TList1_AutoDragRequest(SourceItem As Long, Cancel As Integer)
    If SourceItem = 10 Then
        Cancel = True ' Prevent 10th item from being dragged
    EndIf
End Sub

Private Sub TList1_AutoDragComplete(ByVal SourceControl As TList, ByVal SourceItem As Long, ByVal TargetItem As Long, Operation As Integer, Cancel As Integer)
    If TargetItem = 11 Then
        Cancel = True ' disable dropping operation on 11th item
    EndIf
End Sub
```

#### [See also:](#)

How to Support DragDrop

Manual DragDrop Support

How to Support OLE Drag Drop

### **Manual DragDrop Support**

There are many ways to handle Drag Drop within TList. You may choose to allow the user to drag between two TList controls, within a single TList, or between TList and some other Drag Drop source or destination. When dragging within a TList you may choose to treat the dropped data as a child of the drop target or insert a new item before the drop target. You may wish to restrict the allowed drop targets or drag sources. The choice is up to you. Since Drag Drop is one of the more complicated things you will do with TList, we include several sample applications with the installation kit. The basic technique for dragging and dropping within a TList is as follows:

#### 1. Initiate dragging

You must recognize when to begin dragging within TList. This may be done by setting global variables in the mouse down event to identify the x/y location of the mouse, and checking it again in the mouse move event. When the mouse has moved a certain distance with a depressed left mouse button, initiate dragging by calling the **BeforeDrag** method followed by the **Drag** method.

#### 2. Accepting a Drop

A **DragDrop** event will be triggered. At that time you can identify the **DropTarget** and determine whether to allow dropping. If dropping is allowed you will want to:

- i. use the **ItemBM** property to get the bookmark for the target as the index may potentially change if you will be deleting source items from earlier in the tree.
- ii. use one of the TList.Copy properties to copy source item(s) to a *tree buffer*.
- iii. remove the original source item(s)
- iv. using the bookmark of the target, get its new index and add the items(s) from the *tree buffer* before the target (using the insert property) or as a child of the target (using the add property).

---

Note By default TList updates the **ListIndex** and selections prior to the **MouseDown** event. If you set **MultiSelect** with the intention of to allow dragging of multiple items you should set the **SmartDragDrop** property to postpone this update of the change in selection and **ListIndex**.

---

### See also:

How to Support DragDrop  
Automated DragDrop Support  
How to Support OLE Drag Drop

### ***How to Support OLE Drag Drop***

TList 8 supports OLE style Drag Drop as described in VB 5/6 documentation.

Note that only the **OLEDropMode** property, **OLEDragDrop**, and **OLEDragOver** events are implemented. These provide full support for OLE drop target but do not directly support initiating of OLE dragging from TList. To initiate OLE Drag Drop operations, place another control (any standard control such as a label or textbox supporting OleDragDrop) on the form. This will be used as a surrogate for TList to initiate OleDragDrop. Set that control's visible property to False. Initiate OLE Drag Drop using this hidden control. In the OleStartDrag of the surrogate control, set the data from the source TList control. The following example illustrates the technique:

```
' This example illustrates the use of OLE DragDrop from TList
' using a Textbox as the actual OLE Drag Source
'
' To implement this add a Textbox (Text1) and a TList control (TList1) on
' a form. The Textbox Visible property should be set to False.
'
' Dragging is initiated in the MouseMove event.
' The item under the mouse is copied into the text box,
' then the text box does the OLE drag and drop into the other application.
Option Explicit
Dim XCheck As Single
Dim YCheck As Single

Private Sub Form_Load()
    Dim I As Long
    ' Add some items so we have something to drag
    For I = 0 To 9
        TList1.AddItem "TList Item " & Trim(I)
    Next I
End Sub

Private Sub Text1_OLEStartDrag(Data As DataObject, AllowedEffects As Long)
    ' Copy the data into the other application
    Data.SetData TList1.List(TList1.ListIndex)
    AllowedEffects = 1 ' Copy text
End Sub

Private Sub TList1_MouseDown(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
    XCheck = X
    YCheck = Y
End Sub

Private Sub TList1_MouseMove(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)

    If (Button And 1) And (XCheck > 0) And (YCheck > 0) And _
        ((Abs(XCheck - X) > 50) Or (Abs(YCheck - Y) > 50)) Then
        Text1.OLEDrag ' the text box can initiate OLE Drag and Drop
    End If
End Sub
```

### **Using TList as a DragDrop target from Windows Explorer**

To accept files dragged from Windows Explorer into TList, trap TList's OLEDragDrop event. The TListDataObject passed as a parameter of this event has a **Files** property which provides the names of all files dropped. You can then retrieve these names and display them in TList using the AddItem method.

---

**Note:** You cannot use direct access to IDataObject interface via QueryInterface on TListDataObject interface if you want to support non-standard type of data for OLE drag/drop operations. You should consider using the DataObject property of TListDataObject instead.

---

### See also:

How to Support DragDrop  
Automated DragDrop Support  
Manual DragDrop Support

## How to Sort or Search a Tree

TList supports both Searching and Sorting of the list.

### Searching

Two methods provide searching capabilities:

- FindItem** - Searches based on an item's visible text.
- FindValue** - Searches based on item's associated data.

### Sorting

To sort the children of a given parent item, set the **ItemSorted** property of that item to either 1 (sorting items by displayed text) or 2 (sort items by their associated **Item...Value**) data.

---

**Note** **ItemSorted** is an array property whose Index specifies the item whose children are to be sorted.

---

To sort all root items in the list, specify a parent index of -1.

```
TList1.ItemSorted(-1) = settings%
```

To sort a list by a named associated value (as set with the **ItemValues**) set the **ItemSortingKey** property:

```
TList1.ItemSortingKey(IndexOfTheParentItem%, 0) = "FirstName"
```

Sorting based on arbitrary associated values may be handled numerically or by Ascii character. This is determined by TList depending on what data is being sorted. If associated values were entered as numerical data then 2 comes before 11, if entered as strings, then 11 comes before 2. TList 8 has a new **ItemSortingStyle** property, which gives you more control over the way items are sorted. It even allows to insure placement of parent items (items with children) at the top or bottom of the tree. TList's Grid's **SortingStyle** property is the same as the **ItemSortingStyle** property, but is used to sort Grid rows.

### See also:

TList features and programming techniques

## How to Sort a Grid

There are three 4 properties of a TList Grid object you should be aware of for sorting:

- [TListGrid].Sorted - specifies whether TList should sort the grid. The values are True or False
- [TListGrid].SortingKey - specifies which column should be used as the key for the sorting.
- [TListGrid].SortingStyle - provides additional control over how to treat strings and items with children when sorting, case sensitivity, alpha numeric sorting placing items with children before or after

- [TListGrid].SortingMethod - specifies how the rows of a grid should be sorted.

For example, to use simple AlphaNumeric Sorting.

We'll assume you are not interested in a Case Sensitive Sorting, and that you do not distinguish between rows which have child rows ( a tree)

```
With TList1.Grid
' - Turn off Redrawing of TList while updating sorting parameters
TList.Redraw = False
' - Turn off Sorting of data while updating sorting parameters
.Grid.Sorted = False

'indicate which column you want to use for sorting
'Note that you do not need to have set the ValueNames
'for the different columns - you can work with the default value names
.ValueName = .Coldefs( ColumnNumber ).ValueName
.SortingKey = ValueName

'indicate that you want simple non-case sensitive sorting
.SortingStyle = TL_SORT_IGNORE_CASE

'sort in ascending order
.SortingMethod = 0

'OK, now initiate the sorting according to set parameters
.Sorted = True
'Update the display,
'and allow TList to redraw as required in response to user actions
TList.Redraw = True
End With
```

---

**Note:** there are also properties which may be applied to TList directly such as ItemSortingKey, but these are not generally used when dealing with a Grid, unless you need to sort on hidden values.

---

### See also:

TList features and programming techniques

## How to Access the Clipboard

TList supports copy and paste to the clipboard. You can even copy TList items between TList controls situated in distinct applications.

TList supports its own proprietary format for passing text and associated graphics between TList controls. TList also includes a text presentation of the tree for passing copied items to other applications:

```
TList1.Clipboard(-1) = 0
```

This code will copy contents of the control to the clipboard in both binary and text form. Text presentation can be pasted in any application like WinWord or WordPad.

Example:

```
'Move a selected item from one TList control into another
x% = TList.SellItemIndex(0) 'get index of first selected item
TList_Source.Clipboard(x%) = 0 ' copy item and its children
TList_Source.RemoveItem 19 ' remove item and its children
TList_Dest.Clipboard(5) = 4 'paste before item 5 in destination.
```

See the **Clipboard** and **IsClipboardAvailable** properties and **CopyBuffer** and **PasteBuffer** method descriptions for further information.

---

Note Items pasted from the clipboard do NOT have the same Bookmark as the source from which they are copied. While Bookmarks are Unique identifiers of the original TList item, the data stored in the clipboard is only a copy of the item and its associated data structure.

Note TList clipboard mechanism was designed to provide support for inter-application data exchange. If you want to move pieces of a TList tree inside one application we recommend you use *tree buffer* or File I/O functions and the **Add** or **Insert** TList properties instead of the **Clipboard** property. Use of the clipboard to create new copies of an item with a picture creates a distinct new picture with its own handle and system resource requirements. If you use the *tree buffer* and **Add** or **Insert** properties, TList creates only a new reference to an image, not a distinct new copy.

---

### See also:

TList features and programming techniques

## How to Save and Load Lists - File I/O

TList allows the developer to very quickly and easily save or load a complete tree or branch of a tree, to or from a file. TIP: When working with huge tree's it is well worth saving the Tree to a TList data file and reading that file when initializing the application. This can be significantly faster than executing a large number of **AddItem** statements and associated settings of Fonts, Images, and item Data etc.

TList will also allow the developer to save several trees to the same file, as well as to save or load *tree buffers* to or from a file. You can even mix TList's data with any other data in the same file.

On loading, the tree may be added to the outline at any location. On Saving, TList will save the complete state of the tree together with the Expand/Collapse states.

Of particular importance, TList will optimize the storage of associated images such that an image used by multiple items in the outline is saved only once in the file. TList can save and restore images of any graphic format which are available from Visual Basic (bitmaps, metafiles and even icons).

### Technique

1. Open a file for sequential save or load access. Note that the file should be opened as BINARY only.
2. Optional. Set the **File** property to Read or Write (this creates a TreePictureTable for optimized storage of repeated images). This step is needed only for saving multiple trees to one file. You need to use this property only if you save contents of SEVERAL TList controls, into one file or if you load trees from a file where SEVERAL TList controls have been stored.
3. Set one of the file load/save properties or call one of the file load/save buffer functions:
  - LoadAndAdd** property - loads Tree data from a file, adding it as a subordinate to the item pointed to by the index. (specifying an index of -1 loads to the root)
  - LoadAndInsert** property - loads Tree data from a file, adding it as a peer immediately before the item pointed to by the index.
  - Save** property - saves an item (pointed to by its index) and its subordinates to a file. Specifying an index of -1 will save:
    - the entire contents of the control if the **CurrentIndexMethod** property is set to 0 or 1;
    - the children of the current parent as specified by **CurrentParent** property if the **CurrentIndexMethod** property is set to 2.
  - SaveOne** property - saves to a file an item (pointed to by its index) without its subordinates. Specifying an index of -1 will save all subordinates of the CurrentParent to the file (without their subordinates).

**SaveSub** property - saves the subordinates of an item (pointed to by its index) to a file.

**SaveBuffer** method - saves a *tree buffer* to a file.

**LoadBuffer** method - loads tree data from a file to a *tree buffer*.

4. Repeat step 3 as needed to save or load multiple trees. The same file can be accessed for loading/saving to/from different instances of the TList control.
5. (Optional) Set the File property to Close - this writes the TreePictureTable to the file when saving TList data, or removes unnecessary data when loading TList data. This step is needed only if the File property was previously set to Read or Write to optimize storage of multiple trees in a single file.
6. Close the file.

---

Note Items inserted from a file do NOT have the same Bookmark as the source from which the file was created. While Bookmarks are unique identifiers of the original TList items, the data stored in the file is only a copy of the items and associated data structures.

---

### See also:

TList features and programming techniques

## How to Use Cell / Item Editing

TList provides several mechanisms for powerful in-place editing of data in a List, Tree or Grid.

Techniques range from simple automated editing (just set the EditMode property) to advanced techniques offering customized editing objects such as drop down lists and calendars

Using advanced techniques developers can start, conclude and cancel cell editing manually for the specified cell via **CellEdit** or **ItemEdit** properties. At the same time automatic cell editing that is controlled via **EditMode** property can be initialized by clicking or double clicking mouse on a cell and concluded either by keyboard (ENTER or ESC keys) or by clicking the mouse outside the edited cell).

### See also:

Basic In-Place Text Editing

Use of Built-In Data Editors

TextBox editing

Date/Time editing

Spin editing

Checkbox editing

ComboBox editing

Simulated In-Place Editing Using External Controls

### **Basic In-Place Text Editing**

The **EditMode** property can be used to enable in-place text editing with just a single line of code.

Setting the EditMode enables automatic editing in response to either a single click or double click, for either Tree items, or Grid cells ( optionally including Row and Column titles ).

```
TList.EditMode = EDITMODE_GRID_CELLS + EDITMODE_TREE_ITEMS  
+ EDITMODE_START_ON_CLICK
```

The automated editing process is generally concluded either by keyboard action (ENTER or ESC keys) or by clicking the mouse outside the edited cell.

Developers can also start, conclude and cancel editing manually for a specified Tree / List item or Grid Cell. Setting the **ItemEditText** property (for Tree or List items), or the **CellEdit** property (for a specific Grid Cell) to TLEDITMODE\_BEGIN initiates in-place editing. Setting these properties to TLEDITMODE\_CANCEL or TLEDITMODE\_END concludes the editing of the list item or grid cell.

While no other code is required, there are several events that allow further control over the editing process :

Event Name	Description
<b>RequestEditing</b> <b>GridCellRequestEditing</b>	Triggered before initiating editing of a List or GridCell item These events allow the developer to cancel the process, to specify initial data for editing, and to control certain aspects of the editing window.
<b>AfterEditing</b> <b>GridCellAfterEditing</b>	Triggered immediately after the user finishes or cancels the editing of a List or Grid Cell item, but before the data is updated. These events enables checking edited data before it is accepted to be sure that user provides correct information for the cell data.
<b>EditingKeyDown,</b> <b>EditingKeyUp,</b> <b>EditingKeyPress</b> <b>GridCellEditingKeyDown</b> <b>, GridCellEditingKeyUp,</b> <b>GridCellEditingKeyPress</b>	These events are generated while user is typing and permit processing of the keyboard entry (for example: to disable entering letters for the cell that contains telephone number etc).
<b>ItemEditingChange,</b> <b>GridCellEditingChange</b>	These events are generated in response to end-user editing changes (via keyboard or mouse) that occur while the user is editing grid cells using a built-in editor ( <b>TListTextBox</b> , <b>TListComboBox</b> , <b>TListCheckBox</b> , ....

It is for example possible to force resumption of editing within the **AfterEditing** or **GridCellAfterEditing** Event (for example, after a wrong value is entered by an end user), by setting the **ItemEditText** or **CellEdit** property to TLEDITMODE\_CONTINUE (= 3). You can display a message box from within this event in order to notify users.

### Example

```
Private Sub TList1_GridCellAfterEditing(ByVal vTextToEdit As Variant, vConvertedText As Variant, ByVal objGridCell As
TListProLibCtl.TListGridCell, ByVal CancelledBy As Integer)
    'new text can't be shorter than 5 characters
    If Len(vConvertedText) < 5 Then
        MsgBox "The text size should be at least 5 characters"
        TList1.CellEdit(AnyIndex, AnyIndex) = TLEDITMODE_CONTINUE 'continue editing of this item
    End If
End Sub
```

Editing of a data item is automatically canceled (triggering the **AfterEditing** event) on any of the following occurrences:

- The user clicks on another window or another item of the TList control;
- One of the properties that change the tree structure is set;
- The user resizes control;
- The user scrolls the control.



## Use of Built-In Data Editors

TList supports sophisticated In-Place editing using built-in Data Entry Controls, including Text Editing, Drop Down Listbox Selection, Increment/Decrement Spin Buttons, Checkboxes, and Day/Date/Time Calendar selection. Each editing object has its own properties, and in some cases methods to fine-tune the editing behavior.

Using Checkboxes in Grid Cells (standard and user-defined pictures):

Model	Order	Availability
1.44MB Mitsumi	<input type="checkbox"/>	Out of stock
100MB Zip int	<input checked="" type="checkbox"/>	Out of stock
250MB Zip int	<input checked="" type="checkbox"/>	In stock

Using Checkboxes in Tree Items:

- ☐ Mother Board
- ☒ RAM
  - ☐ DIMM 128MB PC-100
  - ☐ DIMM 128MB PC-133
  - ☐ DIMM 64MB PC-133
- ☐ Video cards
- ☐ HDD
- ☒ Removable Drives
  - ☒ 1.44MB Mitsumi
  - ☒ 100MB Zip int
  - ☒ 250MB Zip int
- ☐ CD-ROM drives
- ☐ Sound Cards

Using ComboBoxes:

Model	Delivery	Guarantee	Comments
1.44MB Mitsumi	2-3 days	months	No comments
100MB Zip int	2-3 days	ee months	Call if out of stock
250MB Zip int	5-7 days	o months	Call before processir



Using Date/Time Controls:

The screenshot shows a software interface with a tree view on the left containing categories: Mother Board, RAM, Video cards, HDD, and Removable Drives. The main area displays a table of hardware components. The 'RAM' section is expanded, showing a list of DIMM modules with columns for Model, Quantity, Last Delivery, and Next Delivery. A calendar control is overlaid on the right side of the table, showing the month of April 2001. The date 4/13/01 is highlighted, and the text 'Today: 4/13/01' is displayed at the bottom of the calendar.

Model	Quantity	Last Delivery	Next Delivery
DIMM 128MB PC-100	136	12/7/99	04 March 00
DIMM 128MB PC-133	136	3/12/01	08 June 01
DIMM 128MB PC-133 HYUNDAI	200	1/15/01	13 April
DIMM 128MB PC-133 Micron	192	2/7/01	
DIMM 128MB PC-133 SEC	168	2/7/01	
DIMM 256MB PC-133	136	1/15/01	
DIMM 256MB PC-133 Hyundai	200	1/15/01	
DIMM 256MB PC-133 Micron	192	10/23/00	
DIMM 32MB PC-100	128	10/20/00	
DIMM 64MB PC-100	128	6/11/00	
DIMM 64MB PC-133	128	2/1/01	

Using Textboxes:

The screenshot shows a software interface with a table of hardware components. The table has columns for Accessory Store, Order, Price, and Comments. The 'RAM' section is expanded, showing a list of DIMM modules. A text box is overlaid on the right side of the table, containing the text 'Use this memory chip for Intel PIII 700 Copermine'.

Accessory Store	Order	Price	Comments
31 Intel PIII 500 Copermine FC-PGA	<input type="checkbox"/>	115\$	
32 Intel PIII 667 Copermine FC-PGA OEM	<input type="checkbox"/>	129\$	
33 Intel PIII 700 Copermine FC-PGA BOX	<input type="checkbox"/>	143\$	
34 Intel PIII 733 Copermine FC-PGA OEM	<input type="checkbox"/>	148\$	
35 Intel PIII 750 Copermine FC-PGA BOX	<input type="checkbox"/>	153\$	
36 Intel PIII 800 Copermine FC-PGA OEM	<input checked="" type="checkbox"/>	174\$	
37 Intel PIII 866EB Copermine FC-PGA OEM	<input type="checkbox"/>	199\$	
38 Intel PIII 933EB Copermine FC-PGA OEM	<input type="checkbox"/>	228\$	
39 Mother Board	<input checked="" type="checkbox"/>		
81 RAM	<input type="checkbox"/>		
82 DIMM 128MB PC-100	<input type="checkbox"/>	41\$	
83 DIMM 128MB PC-133	<input type="checkbox"/>	42\$	
84 DIMM 128MB PC-133 HYUNDAI	<input type="checkbox"/>	46\$	
85 DIMM 128MB PC-133 Micron	<input checked="" type="checkbox"/>	45\$	Use this memory chip for Intel PIII 700 Copermine
86 DIMM 128MB PC-133 SEC	<input type="checkbox"/>	50\$	
87 DIMM 256MB PC-133	<input type="checkbox"/>	78\$	
88 DIMM 256MB PC-133 Hyundai	<input checked="" type="checkbox"/>	89\$	
89 DIMM 256MB PC-133 Micron	<input type="checkbox"/>	85\$	

As with Basic In-Place Text Editing, the editing process may be triggered automatically in response to a click or double click as specified by the TList **EditingMode** or CellDef.EditInfo.**Editable** property, or manually in response to setting the **ItemEditText** or **CellEdit** property. Likewise, **RequestEditing** and **GridCellRequestEditing** events are also triggered when editing begins, although the **Options** property of these events is relevant only to textbox editing

All properties and methods supporting the Editing Objects are collected and organized within the **EditInfo** object property. The **.EditInfo** property applies to any TListCellDef object and thus may be used to specify editing style for the entire list, tree or grid, for a single row, for a grid column, for a grid cell, or for a hierarchic level.

The EditInfo.**Style** property identifies a particular TList Editing object to manage the desired editing behavior (Text editing, Checkbox, DropDown List, etc). Note: You can't access any Editing object until you set the EditInfo.**Style** property to specify the use of the corresponding object.

```
Dim tlCell as TListCellDef
Set tlCell = some TListCellDef object
tlCell.EditInfo.Style = desired Editing Style
```

Valid Style settings **	Editing Object Enabled	Property name referencing Editing Object
TLEDITINFO_TEXTBOX	TListEditBox	.EditInfo. <b>Textbox</b>
TLEDITINFO_CHECKBOX	TListCheckBox	.EditInfo. <b>CheckBox</b>
TLEDITINFO_COMBOBOX	TListComboBox	.EditInfo. <b>ComboBox</b>
TLEDITINFO_DATE_TIME	TListDateTime	.EditInfo. <b>DateTime</b>
TLEDITINFO_SPIN	TListSpin	.EditInfo. <b>Spin</b>

\* \* Bennet-Tec may add to this list in the future. Users requiring new editing objects may also contact Bennet-Tec for customization to meet their requirements.

```

So for instance to specify the use of a combobox in column 2 of a grid
Dim tlCell as TListCellDef
Set tlCell = TList1.Grid.Coldefs( 2 ).CellDef
tlCell.EditInfo.Style = TLEDITINFO_COMBOBOX

Or Alternatively:
TList1.Grid.Coldefs( 2 ).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX

Or Alternatively:
With TList1.Grid.Coldefs( 2 ).CellDef.EditInfo
.Style = TLEDITINFO_COMBOBOX
End With

To reset the Editing style to none - use Set CellDef.EditInfo = "Nothing"
Set TList1.Grid.Coldefs(1).CellDef.EditInfo = Nothing

```

After setting the **EditInfo.Style** property for a cell or cells, additional control over the presentation and edit support behavior may be controlled by properties of the Editing Objects. Each editing object has its own properties, and in some cases methods to fine tune the behavior.

#### See Also:

TextBox editing

Date/Time editing

Spin editing

Checkbox editing

ComboBox editing

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListSpin object, TListDateTime object, TListComboltems object, TListComboltem object

### **TextBox editing**

Setting the Style property of the EditInfo object to *TLEDITINFO\_TEXTBOX* instructs TList to use the TListTextBox object to for end-user editing.

The TListTextBox provides support for setting a Maximum data entry length (see **MaxLength** property), a Minimum and Maximum (see **MinHeight**, **MaxHeight**, and **HeightScale** properties), a minimum and maximum width (see **MinWidth**, **MaxWidth** properties) and automatic adjustment of editing box size (see **Options** property)

```

With TList1.Grid.ColDefs(2).CellDef
.EditInfo.Style = TLEDITINFO_TEXTBOX
.EditInfo.TextBox.Options = TLTEXTBOX_OPT_AUTOWIDTH
.EditInfo.TextBox.MinWidth = 150      'min width of edit window in pixels
.EditInfo.TextBox.MaxWidth = 250     'max width of edit window in pixels
.EditInfo.TextBox.MaxLength = 10     'max number of characters in edit window
End With

```

The following events may also be used to further control or track the text editing process and presentation style of the Editing box:

**RequestEditing, AfterEditing, GridCellRequestEditing, GridCellAfterEditing  
EditingKeyPress, EditingKeyDown, EditingKeyUp  
GridCellEditingKeyPress, GridCellEditingKeyDown, GridCellEditingKeyUp**

#### See Also:

Date/Time editing  
Spin editing  
Checkbox editing  
ComboBox editing

### ***Date/Time editing***

Setting the Style property of the EditInfo object to *TLEDITINFO\_DATE\_TIME* instructs TList to use the **TListDateTime** object for end-user editing of data.

The **TListDateTime** object provides support for setting the earliest and latest valid dates (**Min** and **Max** properties), date display format ( **Format** and **FormatString** properties), and the use of a drop down calendar for date selection (**Options** property)

```

With TList1.Grid.ColDefs(3).CellDef
.EditInfo.Style = TLEDITINFO_DATE_TIME
.EditInfo.DateTime.format = TLFORMAT_LONG_DATE
.EditInfo.datetime.options = TLDATETIME_OPT_CALENDAR
'limit the date range that can be entered
.EditInfo.datetime.Min = cDate("2/01/01")
.EditInfo.datetime.Max = cDate("2/28/01")
End With

```

#### See Also:

TextBox editing  
Spin editing  
Checkbox editing  
ComboBox editing

### ***Spin editing***

Setting the Style property of the EditInfo object to *TLEDITINFO\_SPIN* instructs TList to use the **TListSpin** object for end-user editing.

The **TListSpin** object displays spin buttons during editing allowing the user to increment or decrement the value by a discrete amount (**Step** property) when clicking on the spin buttons. The **Max** and **Min** properties may be used to set a range of values. The **Options** property may be used

to specify placement of increment/decrement buttons, response to arrow keys and the wrapping of the range.

```
With TList1.Grid.ColDefs(3).CellDef
.EditInfo.Spin.Min = 0
.EditInfo.Spin.Max = 100
.EditInfo.Spin.Step = 10
.EditInfo.Spin.Options = TLSPIN_OPT_WRAP _
                        Or TLSPIN_OPT_ARROWKEYS _
                        Or TLSPIN_OPT_HORZ
End With
```

Note that the SPIN edit mode should only be used for Integer values (within a range of -32767 to 32767).

### See Also:

TextBox editing  
Date/Time editing  
Checkbox editing  
ComboBox editing

### **Checkbox editing**

Setting the **Style** property of the **EditInfo** object to *TLEDITINFO\_CHECKBOX* instructs TList to use a **TListCheckBox** object to control the editing / data entry behavior of grid or tree cells.

```
'Use checkbox editing for all cells in column 4 of a grid
Dim tlCell as TListCellDef
Set tlCell = TList.Grid.ColDefs( 4 ).CellDef
tlCell.EditInfo.Style = TLEDITINFO_CHECKBOX
OR
TList.Grid.ColDefs(4).CellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
```

### Checkbox States:

The **TListCheckBox** provides support for both 2 and 3 state checkboxes (as determined by the **States** property). A two state checkbox is either **Checked** or **UnChecked**. A three state checkbox may also be in a **Grayed** (mixed) state. TList's default behavior is to use 2 state checkboxes.

```
'assume tlCell has been defined and points to a valid celldef object
tlCell.EditInfo.Style = TLEDITINFO_CHECKBOX
Dim cBox = TListCheckBox
Set cBox = tlCell.EditInfo.Checkbox
cBox.States = TLCHK_3STATES
OR
TList.Grid.ColDefs(4).CellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
TList.Grid.ColDefs(4).CellDef.EditInfo.Checkbox.States= TLCHK_3STATES
```

For each checkbox state TList automatically displays either a checked, unchecked or grayed checkbox picture. The checkbox may be shown either 2-D or 3-D as specified by the checkbox **Appearance** property (2D – by default).

### Checkbox Images:

It is also possible to specify customized pictures for each state using the **CheckedPicture**, **UnCheckedPicture**, and **GrayedPicture** properties. The placement of the checkbox imaged (above, below, left or right of text) may be controlled using the **Alignment** property of the CellDef

object, or the DefCellAlignment of the TList control itself. The size of the checkbox image (if not the default image) may also be controlled using the PictureWidth and PictureHeight properties of the corresponding CellDef object:

```
'assume tCell has been defined and points to a valid celldef object
'and style has been set to Checkbox
With tCell.EditInfo.CheckBox.
    .CheckedPicture = loadpicture ( "c:\some path\Checked.ico")
    .UncheckedPicture = Picture1.Picture
    .GrayedPicture = TList.MarkPicture(1)
End With
```

### Checkbox Values:

Each Checkbox state has an associated value as specified by the checkbox **CheckedValue**, **UncheckedValue** and **GrayedValue** properties. By default these values are 0 - unchecked, 1 - checked, 2 – grayed (similar to VB checkbox control values).

The state of the checkbox may be determined by reading either the tree list item's **ItemCheckBoxValue**, or the grid cell's **CheckBoxValue** property. When an end user edits the data by checking or unchecking the checkbox, the appropriate value (as defined by **CheckedValue**, **UncheckedValue** or **GrayedValue** properties) is automatically copied to the **CheckBoxValue** property of the grid cell object for the cell in which a checkbox appears, or to the **ItemCheckBoxValue** property for the tree item with a checkbox.

The checkbox state may also be set programmatically by setting the **ItemCheckBoxValue** of the tree item, or the **CheckBoxValue** of a grid cell. With the default values of 0,1,2 for the **UncheckedValue**, **CheckedValue**, and **GrayedValue** properties, setting the TList1.Grid.Cells(Row, Col).CheckBoxValue = 0 unchecks the checkbox, setting to 1 checks the checkbox. Similarly reading CheckBoxValue will return 0,1, 2 depending on the state of the checkbox.

```
' Specify edit settings for checkboxes in Grid cells
' Use checkboxes for all items in Column 4 of the Grid
TList1.Grid.ColDefs(4).CellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
' Set checked/uncheckd state for the check box in the corresponding cell
TList1.Grid.Cells(10, 4).CheckBoxValue = 1 'checks the checkbox
TList1.Grid.Cells(11, 4).CheckBoxValue = 0 'unchecks the checkbox

'Specify edit settings for checkboxes in Tree items
' Use checkboxes for all items in the tree
TList1.DefItemCellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
' Set checked/uncheckd state for the check box in the corresponding item
TList1.ItemCheckBoxValue(2) = 1 'checks the checkbox
TList1.ItemCheckBoxValue(3) = 0 'unchecks the checkbox
'Or to perform the same actions as last two lines of code
'but using TListNode Object
TList1.Node(2).CheckBoxValue = 1 'checks the checkbox
TList1.Node(3).CheckBoxValue = 0 'unchecks the checkbox
```

You can also specify your own values for **UncheckedValue**, **CheckedValue**, and **GrayedValue** properties. String and other types of values may also be used as well. If the **CheckedValue** property is set as the string, "Passed", then setting Grid.Cells(Row, Col).CheckBoxValue = "Passed" sets the checkbox state to Checked and the checked picture is shown.

```
' Specify editing settings for checkboxes in Grid cells
' Use checkboxes for all items in Column 4 of the Grid
With TList1.Grid.ColDefs(4).CellDef
    ' Set Editing Style
    .EditInfo.Style = TLEDITINFO_CHECKBOX
    ' Set values for each possible checkbox state
    .EditInfo.CheckBox.States = TLCHK_3STATES
    .EditInfo.CheckBox.CheckedValue = "Passed"
    .EditInfo.CheckBox.UncheckedValue = "Failed"
    .EditInfo.CheckBox.GrayerValue = "Not Tested"
    .EditInfo.CheckBox.GrayerPicture = LoadPicture ("somepath\NotTested.bmp")
End With
'set the checked/unchecked state of the checkboxes in rows 10, 11 and 12
TList1.Grid.Cells(10,4).CheckBoxValue = "Passed" 'checks the checkbox
TList1.Grid.Cells(11,4).CheckBoxValue = "Failed" 'unchecks the checkbox
TList1.Grid.Cells(11,4).CheckBoxValue = "Not Tested" 'display NotTested bitmap
```

Programmatically setting the grid cell's **CheckBoxValue** or tree item's **ItemCheckBoxValue** property to some value other than that specified by **CheckedValue**, **UncheckedValue**, or **GrayerValue** properties results for a 2-state checkbox in an unchecked state, and for a 3-state checkbox to a grayed state)

Remember, the cell value ( Grid.Cells(r, c).Value) is different from the value corresponding to the Checkbox state ( Grid.Cells(r, c).CheckBoxValue ). The cell value for a cell containing a checkbox is the same as would normally be displayed if there were no checkbox.

#### Checkbox Text:

NOTE: Grid Cells containing checkboxes will have their cell values, TList1.Grid.Cells(Row, Col).Value, displayed next to the checkbox picture (like a caption). For simple Tree items, the value from the **List** property, TList1.List (itemindex), is displayed next to the checkbox. To avoid the display of any text next to the checkbox just set the Value or List property to an empty string:

```
TList1.Grid.Cells(11,4).Value = ""
Or
TList1.List(5)= ""
```

or set the TextAlignment property to hide the text display:

```
'to hide the captions (if any exist) you can use following line of code
TList1.Grid.ColDefs(4).CellDef.TextAlignment = TLTEXTALIGNMENT_NOT_VISIBLE
```

#### Other Checkbox Options:

The **Options** property of the Checkbox object determines whether the checkbox value is changed in response to a single or double click, what visible element (picture or text) must be clicked, and whether the picture corresponding to the Unchecked value is drawn in cells where no state has been set (see **Options** property for details).

#### See Also:

TextBox editing  
Date/Time editing  
Spin editing  
ComboBox editing

### **ComboBox editing**

Setting the **Style** property of the **EditInfo** object to *TLEDITINFO\_COMBOBOX* instructs TList to use a **TListComboBox** object to control the editing / data entry behavior of grid or tree cells.

```
'Use combobox editing for all cells in column 5 of a grid
TList.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
```

The **TListComboBox** object is used for convenient data editing by selecting a value from a list. Its properties and methods manage the values set and the layout/presentation of the combobox window.

A **TListComboBox** control combines the features of a TextBox control and a ListBox control—users can enter information in the text box portion or select an item from the list box portion of the control. The **TListComboBox** may be presented as a DropDown list, or a DropDown List with Text Entry (ability to enter data not in the list)

```
'Use drop-down combobox for all cells in column 5 of a grid
TList.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
TList.Grid.Coldefs(5).CellDef.EditInfo.Combobox.Style = TLCOMBO_DROP_DOWN
Or
'Use drop-down list for all cells in column 5 of a grid
TList.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
TList.Grid.Coldefs(5).CellDef.EditInfo.Combobox.Style = TLCOMBO_DROP_DOWN_LIST
Or
'Use simple combobox for all cells in column 5 of a grid
TList.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
TList.Grid.Coldefs(5).CellDef.EditInfo.Combobox.Style = TLCOMBO_SIMPLE
```

#### Setting Combobox List Items:

To add or delete items in a **TListComboBox** control list, use the **Add**, **AddSafe**, **Remove**, **RemoveItem** or **Clear** methods.

```
' Specify editing settings for ComboBox in Grid cells
' Use checkboxes for all items in Column 5 of the Grid
TList.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
Dim objCombo as TListComboBox
Set objCombo = TList.Grid.Coldefs(5).EditInfo.Combobox
objCombo.Style = TLCOMBO_DROP_DOWN_LIST
objCombo.Items.Add "One"
objCombo.Items.Add "Two"
objCombo.Items.Add "Three"
'now enable automatic editing for the grid cells
TList1.EdittingMode = EDITMODE_GRID_CELLS
' The end-user can now double click on a cell containing a combobox
' and select any value using the drop-down list.
' This value will be set to the corresponding cell's value
' (TList1.Grid.Cells(...).Value.Value property).
```

#### Formatting Combobox List Items:

Combobox list items by default inherit their format (Font, Color, TextAlignment , etc, . . . ) from the cell which is being edited. Each combobox list item may however be distinctly formatted. In this case when a list item is selected from the combobox, the end-user will see the resulting cell formatted in this way. The underlying cell's celldef formatting properties are not changed however, they are just superseded.

```
With TList1.Grid.Cells(7,5).celldef
' set the background color for the cell to Black
' by default all items in the list will have this background color
.BackColor = RGB ( 0,0,0)
```



```
.EditInfo.ComboBox.AddItem "Excellent"
.EditInfo.ComboBox.AddItem "Satisfactory"
' add an item to the combobox list with a background color of RED
Dim objTLComboItem as TListComboItems
Set objTLComboItem = .EditInfo.ComboBox.AddItem "Warning"
ObjTLComboItem.BackColor = RGB (255, 0, 0)
' after selecting this item from combobox the cell background will be Red
End With
```

### Using the TListComboBox for Intelligent Formatting:

The **TListComboBox** object can also be used to design a more convenient user interface, presenting meaningful textual data to the user, while maintaining possibly more useful numeric data representation for programmatic manipulation. For example:

```
' Specify editing settings for ComboBox in Column 5 of a Grid
' assume the same initialization section as in previous sample code
' But now we create association between actual values
' and their visible representation for end-user using TListComboBox object
With TList1.Grid.ColDefs(5).CellDef.EditInfo.ComboBox.Items
.Add 1, , "Hour" ' user sees "Hour" representing Value of 1
.Add 24, , "Day" ' user sees "Day" representing Value of 24
.Add 168, , "Week" ' user sees "Week" representing Value of 168
End With
' The underlying data can still be numbers
TList1.Grid.Cells(7, 5).Value = 1
TList1.Grid.Cells(8, 5).Value = 24
TList1.Grid.Cells(9, 5).Value = 168
' NOTE: The user will see the symbolic representation of the data
' in the cell, NOT the underlying numbers.
' eg: the user will see the word "Day" instead of the number "24"
```

**NOTE:** The TListComboBox object can be used to change the representation of data in this way, even while disabling editing for some particular cells (columns, rows etc) using the EditInfo.**Editable** property. In this case the corresponding representation from the TListComboBox list will be displayed rather than the actual cell data – the end-user will see the text (or picture) representation but he won't be able to edit such cells.

The TListComboBox object provides even more ways to control the appearance of cells – automatically changing the background and foreground colors, picture, gradient, alignment, of a cell depending on the data that the cell contains. The next example demonstrates how to automatically set the background color to red for all the grid cells containing the string "Apples" and to green for the grid cells containing "Pepper":

```
'Set up a Tree item and an ItemGrid under it containing 6 rows and 5 columns
TList1.AddItem "Root"
With TList1.ItemGrid(0)
.Cols = 5
.Rows = 6
' Specify the actual data, as either "Apples" or "Peppers"
For row = 1 To 5
For col = 1 To 4
.Cells(row, col).Value.Value = _
IIf( (row + col) Mod 2 = 1, "Apples", "Peppers" )
Next col
Next row

' Set the EditStyle for all cells in the ItemGrid to ComboBox
' By associating combobox values with background colors,
' the ComboBox will be used by TList to automatically format the data
With .GridCellDef.EditInfo
.Style = TLEDITINFO_COMBOBOX
```



```
' Add ComboBox Item with value of "Apples" and backcolor Red
.ComboBox.Items.Add("Apples").BackColor = RGB(255, 0, 0)
' Add ComboBox Item with value of "Peppers" and backcolor Green
.ComboBox.Items.Add("Pepper").BackColor = RGB(0, 255, 0)

End With
```

End With

### See Also:

TextBox editing  
Date/Time editing  
Spin editing  
Checkbox editing

### ***Simulated In-Place Editing / TList as a Container***

TList can accept Child controls, with TList acting as a standard container or Simple Frame. This allows simulation of editing cell content with an arbitrary control. One can use TList properties to get coordinates of the active cell in the control, position a child control using these coordinates, get data from the cell, set child control's properties, retrieve the modified data after editing and reset the cell's properties.

See sample project 15 for details on use.

### ***Additional Notes on In-Place Editing***

#### Editing Virtual Items:

Virtual Items may be edited, but the edited string will be stored in a virtual item only as long as this item exists in memory. Once TList removes a virtual item from memory, the data will be lost. It may be reasonable therefore to save that data in the original data source during the **AfterEditing** event.

#### Editing Selected Items:

When editing selected items, TList's default behavior is to draw a border around the item and set colors as if the rectangle were a standard textbox. Thus Foreground/Background colors for the item being edited may change during the editing operation - colors will be restored upon completion of editing.

### **How to Use Bookmarks**

TList supports bookmarking of items.

An ItemBookmark, as contained in the **ItemBM** array property, is a long integer pointing to an item. The bookmark associated with an item will not change even if the item's index changes as a result of adding items higher in the list or changing the **CurrentIndexMethod**. The **Add** and **Insert** properties have also been extended to support working with Bookmarks as they already had worked with *tree buffers*.

---

#### Notes

- a) Bookmarks are independent of the actual instance of a control and so may be useful in copying items between two distinct TList controls using **Add** and **Insert** properties.
  - b) A bookmark becomes invalid as soon as the item to which it refers is deleted. When a property is set to an invalid bookmark an error, "Invalid bookmark", is generated. Also, bookmark information is **NOT** stored when items are saved to a file, copied to the clipboard or copied to a *tree buffer*.
  - c) Note Bookmarks are also NOT preserved when copying to the clipboard, copying to a TreeBuffer or saving to a TLT file.
-

For further information, refer to descriptions of the **ItemBM**, **Add**, **CurrentItemBM**, **ItemParentBM** properties and the **IsValidBM** and **IndexByBM** methods.

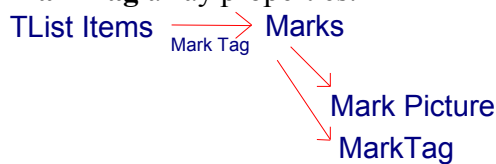
**See also:**

TList features and programming techniques

## How to Assign Categories - TList Mark Support

TList provides support for defining categories of items - for instance: Editable items, Group Managers, Out of Stock Parts.

Each Item has an **ItemMark** property defining the category (0 to 255) to which it belongs. Each Category/Mark has an associated image and string tag specified by elements of the **MarkPicture** and **MarkTag** array properties.



To display the *MarkPictures* set the **ViewStyleEx** property to 2 or 3. The *MarkPictures* will be shown on the far left of the control for each item belonging to a Mark Category. Updating the **ItemMark** will update the display to show the appropriate image for that item based on its new category. Updating a **MarkPicture** will update the display to show the new image for all items in the associated category.

---

Note Item Marks can be employed to hide or display whole categories of items. Setting **MarkedItemsAlwaysHidden**(markindex) will hide all items having that mark index assigned to their **ItemMark** property (assuming **ShowHiddenItems** is False). This can be used to display different components of a large tree to different audiences.

---

Just as the **ItemValues.Value** properties are useful for storing item specific data for each item, the **ItemMark** is useful to assign data from a category to individual TList items. Also changing the data in the **MarkTag** and **MarkPicture** arrays is much easier and faster than changing the Image property or **ItemValues.Value** property for each associated element of the TList control.

Example:

```
Sub TList_Click()  
    'Get some text from Mark Array  
    '    based on which item is clicked  
    Item_Clicked% = TList.ListIndex  
    Tagnumber% = TList1.ItemMark(Item_Clicked%)  
    GetText$ = TList1.MarkTag(TagNumber)  
End Sub
```

For further information refer to the descriptions of the **ItemMark**, **MarkPicture**, **MarkTag**, **MarkHeight**, **MarkWidth**, **MarkedItemsAlwaysHidden**, and **ViewStyleEx** properties and **MarkClick** and **MarkDbClick** events.

**See also:**

TList features and programming techniques

## How to Control the Display of Plus/Minus Pictures

The appearance of plus/minus pictures next to items is determined by a combination of the **ViewStyleEx** property, the **ItemPMPicType** array property, whether the item has any children, and whether it is expanded. This is summarized in the following table:

ItemPMPicType	ViewStyleEx	Item has Children	Item is collapsed	Image is displayed
0 (default)	other than 1 or 2			None
0 (default)	= 1 or 2	False		None
0 (default)	= 1 or 2	True	True	Minus (-)
0 (default)	= 1 or 2	True	False	Plus (+)
1				Plus (+)
2				Minus (-)
3				None

The **ViewStyleEx** setting determines whether to show plus/minus pictures (depending on expand/collapse state of the item) for those items which have subordinate children. Setting **ViewStyleEx** to a value other than 1 or 2 indicates that no plus/minus image should be shown. Setting **ViewStyleEx** to either 1 or 2 indicates that the display of a plus/minus image should be determined based on the expand collapse state of items with children. No plus/minus image is displayed for items without children.

This overall behavior of the tree may be overridden by the **ItemPMPicType** property array settings for each individual item. For example:

```
'a minus picture will be displayed next to the 25th item
TList1.ItemPMPicType(25) = 2
```

For further information refer to the description of the **ViewStyleEx** and **ItemPMPicType** properties.

#### See also:

TList features and programming techniques

## How to Upgrade an Old TList 3/Pro OCX Project to Use New TList OCX

To make the conversion from earlier versions of TList to TList 8 as simple as possible, the TList 8 installation kit includes a conversion utility. This can be accessed from the Start Menu group.

1. First make a backup of your original project
2. Next run the conversion utility and specify the name and location of the project being converted. The converter will replace all instances of the TList 3/Pro OCX on your forms with TList 8.
3. Open your project and replace the TLO30P.BAS file with TLIST8.BAS
4. The only code you should need to change is where you have made reference to an obsoleted property: in particular the Titles property is no longer supported; use TList grid items and column titles instead. Other obsolete properties include **NoIntegralHeight**, **CoerceIndex** and **BackwardCompatible**.

#### See also:

TList features and programming techniques

## How to Upgrade an Old VBX Based Project to Use TList OCX in Place of the VBX

The following procedure may be used to upgrade your old project:

1. Load your old Visual Basic 3.0 project into Visual Basic 4.0 or 5.0 environment. You will be asked whether to convert TLIST.VBX or not. After successful upgrade the TList VBX will be substituted with the OCX version of TList.
2. Next, replace the old TList constants and function declarations file TL30.BAS or TLIST.TXT file, with the OCX version TLIST8.BAS file. Be sure that you have replaced this file, otherwise you may have some problems.
3. If you use TList as Drag Drop source in your project you need to do as follows:
  - Call the **OnDragOver** method in the first line of your **DragOver** event:

```
Private Sub TList1_DragOver(Source As Control, _
    X As Single, Y As Single, State As Integer)
    TList1.OnDragOver X, Y, State
    ...
End Sub
```

- Call the **OnDragDrop** method in the first line of your **DragDrop** event:

```
Private Sub TList1_DragDrop(Source As Control, _
    X As Single, Y As Single)
    TList1.OnDragDrop X, Y
    ...
End Sub
```

- Call the **BeforeDrag** method before the call to **Drag** method, which initiates the Drag Drop:

```
TList1.BeforeDrag
TList1.Drag 1
```

In order to increase the number of items which can be held by TList, we changed the data type of TList indexes to Long values. For this reason, parameters of some properties, events and methods have been changed. Below is the list of changes which you should repeat in your project:

Old declaration	New declaration (parameters referencing item indexes are now LONG values)
Sub TList1_Expand(I As Integer)	Sub TList1_Expand(ByVal I As Long)
Sub TList1_Collapse(I As Integer)	Sub TList1_Collapse(ByVal I As Long)
Sub TList1_PlusMinusClick(I As Integer)	Sub TList1_PlusMinusClick( ➡ ByVal I As Long)
Sub TList1_PlusMinusDbClick(I As Integer)	Sub TList1_PlusMinusDbClick( ➡ ByVal I As Long)
Sub TList1_AfterEditing( ➡ ItemIndex As Integer, ➡ EditedText As String, ➡ CancelledBy As Integer)	Sub TList1_AfterEditing( ➡ ByVal ItemIndex As Long, ➡ EditedText As String, ➡ ByVal CancelledBy As Integer)
Sub TList1_RequestEditing ( ➡ Cancel As Integer, ➡ ItemIndex As Integer, ➡ TextToEdit As String, ➡ Options As Integer)	Sub TList1_RequestEditing ( ➡ Cancel As Integer, ➡ ByVal ItemIndex As Long, ➡ TextToEdit As String, ➡ Options As Integer)
Sub TList1_EditingKeyDown ( ➡ ItemIndex As Integer, ➡ KeyCode As Integer, Shift As Integer)	Sub TList1_EditingKeyDown ( ➡ ByVal ItemIndex As Long, ➡ KeyCode As Integer, Shift As Integer)

	Integer)
Sub TList1_EditingKeyPress( ➡ ItemIndex As Integer, ➡ KeyAscii As Integer)	Sub TList1_EditingKeyPress( ➡ ByVal ItemIndex As Long, ➡ KeyAscii As Integer)
Sub TList1_EditingKeyUp ( ➡ ItemIndex As Integer, ➡ KeyCode As Integer, Shift As Integer)	Sub TList1_EditingKeyUp ( ➡ ByVal ItemIndex As Long, ➡ KeyCode As Integer, Shift As Integer)
Sub TList1_MarkClick (I As Integer)	Sub TList1_MarkClick (ByVal I As Long)
Sub TList1_MarkDbClick (I As Integer)	Sub TList1_MarkDbClick (ByVal I As Long)

**See also:**

TList features and programming techniques

## How To Detect the Version Number Of TList

To determine which version of TList you are using:

- At design-time - use the **About** property which shows a dialog with the current version and date stamp.
- At run-time - use the Version property, which returns a Long with minor and major versions of the control.

**See also:**

TList features and programming techniques

## How to Trap Right Mouse Clicks

To simplify right-mouse menu implementation an **ItemClick** event has been added. This event provides a Button parameter to identify which mouse button has been clicked by end-user.

**See also:**

TList features and programming techniques

## How to Navigate a Web Site with TList

Your site is becoming too complex? You don't know how to paste together tons of pages to make them look organized? Would you like to make your Web Site content easily manageable and transparent to the user?

The easiest way to do that is to follow a proven Outline / Viewer approach. By using Frames in your Web site and placing the TList outline control into one frame on the left side and your actual content on the right, your users will be able to understand your organization at a glance and jump to the desired location with a single click.

Using TList you can easily create such pages. Also you don't have to insert references to all pages of your Web Site into the outline, just select the ones you think are reasonable to expose.

Use the following examples to modify your site:

- Look at our Web Site built exactly with that technique at <http://www.Bennet-Tec.Com>. **(Select the Internet Explorer View.)**
- Try our *Samples for Internet Explorer*, which are in the TList program group. Sources are available in the TList Installation Directory\Samples directory.

- If you have a Microsoft Control Pad application use it to insert a TList control into your HTM document. If you don't, you can insert it manually, just paste the following lines in your HTML file:

```
<OBJECT
ID="TList1" WIDTH=100% HEIGHT=100%
CLASSID="CLSID:A3FC1700-924C-11D5-8FE5-0004ACD846EA"
CODEBASE="http://www.btis.com/activex/TLIST8.cab#Version=8,0,10"
DATA="http://{your server address}/FileGeneratedWithTDesigner.TLT"
>
</OBJECT>
```

CODEBASE refers to the site where the TLIST8.CAB file can be found. The Bennet-Tec download area always has the very latest update for the ActiveX control. Make sure you let us know about your Web site so we can reach you if this location should ever change.

DATA refers to a file with TList control content. No programming is required to build the data file. This file is generated by the TDesigner application and must be accessible by your Web Site users; that is, you must put it in a subdirectory of the WWWRoot directory.

- Save your .HTM document.
- Start the TDesigner application and generate a .TLT file with desired TList content.
- Put .HTM and .TLT files on your Web server and test the whole thing.
- Test your Web site before exposing it to other users.

#### See also:

TList features and programming techniques

## How to Print With TList

TList provides a powerful print engine.

There are two ways to print with TList – using the PrintOneStep method, and using the TList Report object.

A) The PrintOneStep method is a simple but very flexible interface providing the ability to print TList content in one step with just one call.

This method is recommended for use in almost all cases, as it provides support for the vast majority of printing needs.

The following procedure illustrates this procedure:

```
Dim OutputDC as Variant
Dim StartItem As Long, EndItem As Long
Dim PrintOptions As Long, Zoom as Integer

OutputDC = -2 'TList will ask user to select printer
StartItem = 0
EndItem = -1 ' print all items
PrintOptions = TL_PRNOS_BACKGROUND Or TL_PRNOS_ABORT_STYLE ' TList will print background and display abort dialog
Zoom = 100 ' zoom factor in percent

Call PrintOneStep(OutputDC, 100, 100, 100, 100, StartItem, EndItem, PrintOptions, "", "", Zoom)
```

B) An advanced printing interface is also provided using the special TList.Report object to give full control over the printed report structure. This use of the Report object method may provide some

additional flexibility in rare instances but is considered obsolete and not recommended due to it's complexity.

There are basically 4 steps involved in printing with the Report object:

- 1) Initialize the printer
- 2) Set up the TList Report Object
- 3) Send the TList Report Object to the printer
- 4) Close the Printer Object

The following procedure illustrates printing with the Report object:

1. Initialize Printing

If printing to a physical device, Initialize the printer **Printer** object manually before any call is made to the **Report** object:

```
Printer.Print ""
```

2. Set up the TList Report object

a. set the destination device context

If you want to print to a printer pass this **Printer** object to TList,

If you want to print to an arbitrary DC, for example if you want to use a picturebox for a Print Preview destination, pass this DC to TList instead of the **Printer** object:

```
Set TList1.Report.PrinterObject = Printer  
Or  
TList1.Report.PrinterObject = PictureBox.HDC
```

b. Specify the range of items to be printed:

```
TList1.Report.FirstItem = 0  
TList1.Report.LastItem = -1 ' -1 means "up to the last item"
```

c. Specify margins for all pages (in twips):

```
TList1.Report.LeftMargin = 100  
TList1.Report.TopMargin = 100  
TList1.Report.RightMargin = 100  
TList1.Report.BottomMargin = 100
```

d. Specify how the printing progress will be displayed to the end user:

```
TList1.Report.AbortWindowStyle = 1 'show internal TList's progress dialog
```

e. Tell TList to go to a new page automatically when there is a need:

```
TList1.Report.AutoNewPage = TRUE
```

f. Now call the PrepareForPrinting method which instructs TList to repaginate all the pages and create a Pages collection, during this operation it is possible to change settings for each page separately handling the **PreparePage** event: TList1.Report.PrepareForPrinting

g. While **PrepareForPrinting** is in progress, TList generates a sequence of **PreparePage** events. You can handle this event to control the **Pages** collection creation. For example, to print all odd TList pages on the left side of the page, and even pages on the right side, put the following code inside the **PreparePage** event:

```
Private Sub PreparePage(PrinterObject As Variant, _  
    ReportPage As TListReportPage)  
If (ReportPage.PageNumber Mod 2) = 0 Then  
    ReportPage.LeftMargin = 100  
    ReportPage.RightMargin = PrinterObject.Width/2+100  
Else  
    ReportPage.LeftMargin = PrinterObject.Width/2+100  
    ReportPage.RightMargin = 100  
End IfEnd Sub
```

---

**Note It is not possible to change properties for each page outside the PreparePage event.**

---

After completion of the PreparePage event the **TList1.Report.Pages** collection contains a formatted TList report. Now TList is ready to actually print something:

### 3) Send the TList Report to the printer

```
TList1.Report.Print 'calling this method without parameters forces TList to print all pages  
' Or to print a range of pages use the following syntax:  
TList1.Report.Print 2, 3 'In this case TList prints only the 2nd and the 3rd pages
```

While printing is in progress, TList:

- shows a dialog box with a progress indicator
- generates the **BeginPage** and the **EndPage** events, which can be used to print additional information on each page (if needed).

```
4. To complete the printing process, close the Printer object :Printer.EndDoc  
Set TList1.Report.PrinterObject = Null
```

#### [See also:](#)

TList features and programming techniques

## How to Use TList's RTF Support

TList items and cells may be fully formatted, accepting standard RTF formatting including character and paragraph formatting, different colors, superscripts and subscripts, embedded pictures, embedded OLE objects etc. This feature is of great value in highlighting key words such as search term results or syntax keywords within a list.

The **RTFStyle** property determines whether a text string should be interpreted and displayed as RTF text or as simple ASCII text. **RTFStyle** may be applied to the entire tree, or to any hierarchical level or any column, thus plain ASCII text and RTF formatted text may be mixed within the same tree.

To turn on the RTF formatting for the whole tree use the **DefItemCellDef** property:

```
TList1.DefItemCellDef.RTFStyle = 1
```

To turn on the RTF formatting just for one column use the following code:

```
TList1.ItemGrid(ItemIndex&).ColDefs(ColumnNumber&).CellDef.RTFStyle = 1  
' Or  
TList1.Grid.ColDefs(ColumnNumber&).CellDef.RTFStyle = 1
```

Then assign the RTF formatted text to the **List** or the **Text** property.

```
RTFFormattedString = ...RichTextBox1.TextRTF  
TList1.AddItem RTFFormattedString  
' Or  
TList1.List(ItemIndex&) = RTFFormattedString  
' Or  
RTFFormattedString = ..."{\rtf1 is it \b1 bold \b0 or \i1 italic?}"  
TList1.Grid.Cells(row,column).Value = RTFFormattedString
```

---

Note: The rules for constructing RTF formatting strings in TList are the same as for the MS Rich Textbox. To debug any problems you may have, try displaying the same string in the MS RichText box.

---

#### [See also:](#)

TList features and programming techniques

## How to Use the VisualRoot Property

TList's VisualRoot property allows display of a restricted portion of the tree, basically presenting the end-user with a subset of the tree. This may be used for showing different branches to different users, or to provide restricted access to higher levels of the tree based on an end-user's permissioning. For example, let's assume that we have the tree as shown below:

```
Item 1  
    Item 1.1  
    Item 1.2  
        Item 1.2.1
```



```

        Item 1.2.2
    Item 1.3
Item 2
Item 3

```

To make TList display only children of Item 1 we have to set VisualRoot property as follows:

```
TList1.VisualRoot = 0 ' TList indices start with 0 as the the index of the first Item
```

Now TList displays the tree as follows:

```

Item 1.1
Item 1.2
    Item 1.2.1
    Item 1.2.2
Item 1.3

```

To display the whole tree just set the VisualRoot property to -1:

```
TList1.VisualRoot = -1
```

The Visual Basic sample project "*VisualRoot Demo*" is included in the installation kit as an example.

[See also:](#)

TList features and programming techniques

## How to Use LevelDefs for Sorting

TList allows the developer to specify sorting settings for all items belonging specified level. This functionality is available via LevelDef object interface (SortingKey, SortingStyle, SortingMethod). The developer can specify all necessary sorting parameters using corresponding LevelDef object and TList automatically applies these settings on all items at the level.

---

**Note:** By default TList applies LevelDef sorting settings only for already loaded items. So after adding new items at some level it is necessary to resort corresponding level manually turning Sorted property on/off. This default behavior may be controlled by setting or resetting the **TL\_MOD\_SORT\_LEVELS** flag in the Modifications property. HOWEVER - with LevelDef.Sorting automatically applied as items are added, operations such as adding items will be slower. So to speed up adding items and other tree modifications (cut/paste, move, drag/drop, etc) it is suggested to turn off this updating feature in the Modifications property and then to manually resort the whole tree structure (set LevelDef.Sorting again) after completing all adding/copy/paste operations.

---

### Example

```

'populate TList with items
Call CreateTListTree(TList1)

'specify sorting attributes
TList1.LevelDefs(1).SortingStyle = TL_B_SORT_IGNORE_CASE
TList1.LevelDefs(1).SortingMethod = 0 'sort in the ascending order
TList1.LevelDefs(1).Sorted = True 'start sorting

'turn off automatic application of sorting settings for new items
TList1.Modifications = TList1.Modifications Or TL_MOD_SORT_LEVELS

'Add more items ' LevelDef sorting will not be applied to new level 1 items
Call GetNewItem(TList1)

'Reset Sorting for Level 1 items
TList1.LevelDefs(1).Sorted = False 'first turn sorting off
TList1.LevelDefs(1).Sorted = True 'then resort items again

```

[See also:](#)

TList features and programming techniques

## How to Use TListNodes and TListNode Objects

**TListNodes** collections and **TListNode Objects** provide the developer with a convenient way of accessing and manipulating items in the tree.

Using **TListNodes** interface a developer can easily add, remove and modify tree structure without referencing TList itself. This allows manipulating the tree structure in an object-oriented way. For example it can be convenient to pass a reference to the some specified **TListNodes Object** inside the procedure and manipulate with it without accessing the TList object.

```
Sub AddRootItems(RootNodes As TListNodes)
    Dim CurNode As TListNode
    Dim Index As Long
    For Index=1 To 10
        ' The Add method of a TListNode Object returns a reference to the newly added node
        Set CurNode = RootNodes.Add("Root"& Str(Index), -1, TLNODERELATION_CHILD)
        ' Pass the new root as a node to a subroutine which adds child nodes
        Call AddSubItems(CurNode)
    Next Index
End Sub
```

Using **TListNode** interface it is possible to get a reference to the parent item, next and previous siblings to add a new subordinate item. Also this interface provides an access to all item's properties for manipulating item's appearance and functionality.

```
Sub AddSubItems(ParentNode As TListNode)
    ' Add child nodes to whatever node is passed to this routine
    Dim CurNode As TListNode
    Dim Index As Long
    For Index=1 To 5
        Set CurNode = ParentNode.Add("Child" & Str(Index), TLNODERELATION_CHILD)
        CurNode.Image = LoadPicture()
        CurNode.SelectedImage = LoadPicture()
        CurNode.BackColor = RGB(0, 255, 0)
        CurNode.Tag = Index
    Next Index
End Sub
```

### See also:

TList features and programming techniques

## How to Use IntelliMouse Functionality

TList supports IntelliMouse and the developer can take advantage of additional navigational features in the control. Note: the IntelliMouse is a pointing device with a wheel control between the two buttons. The wheel control is both a wheel and a button that makes common navigation functions easy to perform with the mouse. So a user can rotate the wheel to scroll through a window more quickly than using the scroll bars. TList provides the developer an enhanced control over this functionality with the **WheelScrolling** property. Thus it is possible to use either the default scrolling behavior or to handle the behavior manually by corresponding **MouseWheel** event. By default IntelliMouse support is turned on and users can scroll TList's trees by Line by Page or from one Root to the next using corresponding wheel/key combination (wheel scrolls+NONE/CTRL/SHIFT key).

Note: TDesigner application provides access to modify IntelliMouse support behavior.

### See also:

TList features and programming techniques

## How To Resize Rows And Columns

There are two methods to force a column to change its width according to length of the data contained in it.

A) Call the `AutoSizeColumn` method in code.

For example:

```
' Adjust the width of the 3rd column
TList1.Grid.AutoSizeColumn 3, GRID_AUTOSIZECOL_DEFAULT
```

B) Double-click on a the column header separator at the right of the column.

Note that in order to enable this feature ( which is turned off by default ) it is necessary to set the `AutoSize` property to include the flag bit,

`GRID_AUTOSIZE_COLUMN_ON_SEPARATOR_DOUBLECLICK`

For example

```
TList.AutoSizeOptions = GRID_AUTOSIZE_UNLIMITED_WIDTH OR
GRID_AUTOSIZE_COLUMN_ON_SEPARATOR_DOUBLECLICK
```

### See also:

TList features and programming techniques

`AutoSizeOptions` property

`AutoSizeColumn` method

`AutoSizeRow` method

## How to Use ToolTips

TList can automatically display *tooltips*, showing the full text of an item, when the mouse cursor is moved over an item clipped by the borders of the control. Full control over the style and color of the ToolTip presentation is provided via the `ToolTipsText`, `ToolTipsMode`, `ToolTipsVisualStyle`, `ToolTipsForeColor`, `ToolTipsDelay`, and `ToolTipsBackColor` properties.

In addition full control over the content and appearance of tooltips for individual list/tree nodes or grid cells is provided through the new `ItemToolTip` and `ToolTip` properties, and the new objects: `TListShowTooltipArgs`, `TListToolTip`, `TListTooltipStyle` and `TListTooltipWindow`.

Lastly, a new `ShowToolTip` event was introduced with an `objToolTip` parameter to provide additional control at the time of the tooltip display.

### Example

```
Sub Form_Load()
    TList1.ToolTipsMode = TLTOOLTIPSMODE_ENABLE
    For i = 1 to 100:
        TList1.AddItem "Item " & str ( i )
        TList1.ItemToolTip(-3).Text = "Tooltip for item" & str ( i )
    Next i
End Sub
```

### See also:

TList features and programming techniques

## How to Copy from one TList to Another

TList makes it very easy to quickly copy items within a single instance of TList or from one TList control to another

One technique is to use the Clipboard ( see section "How To Access the Clipboard") however it is also easy to use TList's internal buffers for copying items.

Here is an example showing how to copy all items from one TList control to another

```
' Copy all items from 1st TList into a TreeBuffer
Dim tree_buffer&
tree_buffer& = TList1.CopyItem (-1)
' Clear 2nd TList and then add all items from buffer into 2nd TList
TList2.Clear
TList2.Add(-1) = tree_buffer&
' Release memory associated with the tree buffer
TList1.FreeBuffer tree_buffer&
```

This is VERY FAST

### See also:

TList features and programming techniques

## How to Smoothly Scroll Tall Items

TList 8 now provides unique support for very tall items within a list / tree / grid, such as when presenting large images, or long memos.

Unlike most Lists, Tree's, Grids, and unlike previous editions of TList itself, all of which scroll only by whole rows and therefore never provide access to the complete content of a data cell which is taller than the control itself ( such as a large picture or very long memo ), TList 8 provides support for scrolling in either a row-by-row manner or in a smooth pixel based manner. Moreover TList 8 can be set to automatically recognize tall cells and intelligently adjust the scrolling mechanism in response to the height of the data currently in view.

This vertical scrolling support is implemented through the following properties:

- **ScrollVModeKeyboard** - controls TList's vertical scrolling behavior in response to keyboard Up/Down arrow keys
- **ScrollVModeMouse** - controls TList's vertical scrolling behavior in response to mouse clicks upon the Up/Down buttons within the vertical scrollbar.
- **ScrollVStepMouse** - controls the distance by which TList scrolls vertically when smooth scrolling in response to clicks on Up / Down buttons within TList scrollbar.
- **ScrollVStepKeyboard** - controls the distance by which TList scrolls vertically when smooth scrolling in response to keyboard Up/Down arrow keys

## How to Support Excel Style Navigation While Editing

TList 8 provides direct support for a user to navigate from cell to cell with arrow or tab keys while editing. So if user is in process of editing one cell, and hits tab key, the editing of that cell will be completed, the next cell will become active, and the user may immediately continue editing of this newly active cell.

This is implemented with the following new properties:

- TList.**KeyboardNavigateWhileEditing**
- TListEditInfo.**EditableStartOptions**

Here is an example to illustrate the setup

```
With TList1.Grid
  .Cols = 4
  .Rows = 5
  .ActivationMode = TL_ACTIVMODE_CELL
  .SelectionMode = TL_SELMODE_SINGLE
  With .GridCellDef
    .Selectable = TL_SEL_DISABLED
    With .EditInfo
      .Editable = TLEDITABLE_ALWAYS_ONACTIVATE
      .KeyboardNavigateWhileEditing = _
        TL_EDITNAVIGATION_ON_ARROW_KEYS _
        OR TL_EDITNAVIGATION_ON_TAB_KEY
    End With
  End With
End With
```

---

## TList Diagrams (Objects, Functionality And Relationships)

The following diagrams show the relationships between objects exposed by TList. They explain how to access TList tree and grid structures holding data and formatting.

Features Diagram

Tree Data/Objects Diagram

Grid Data/Objects Diagram

Objects/Relationship Diagram

## Features Diagram

The following diagram illustrates how to access key functionality features of TList.

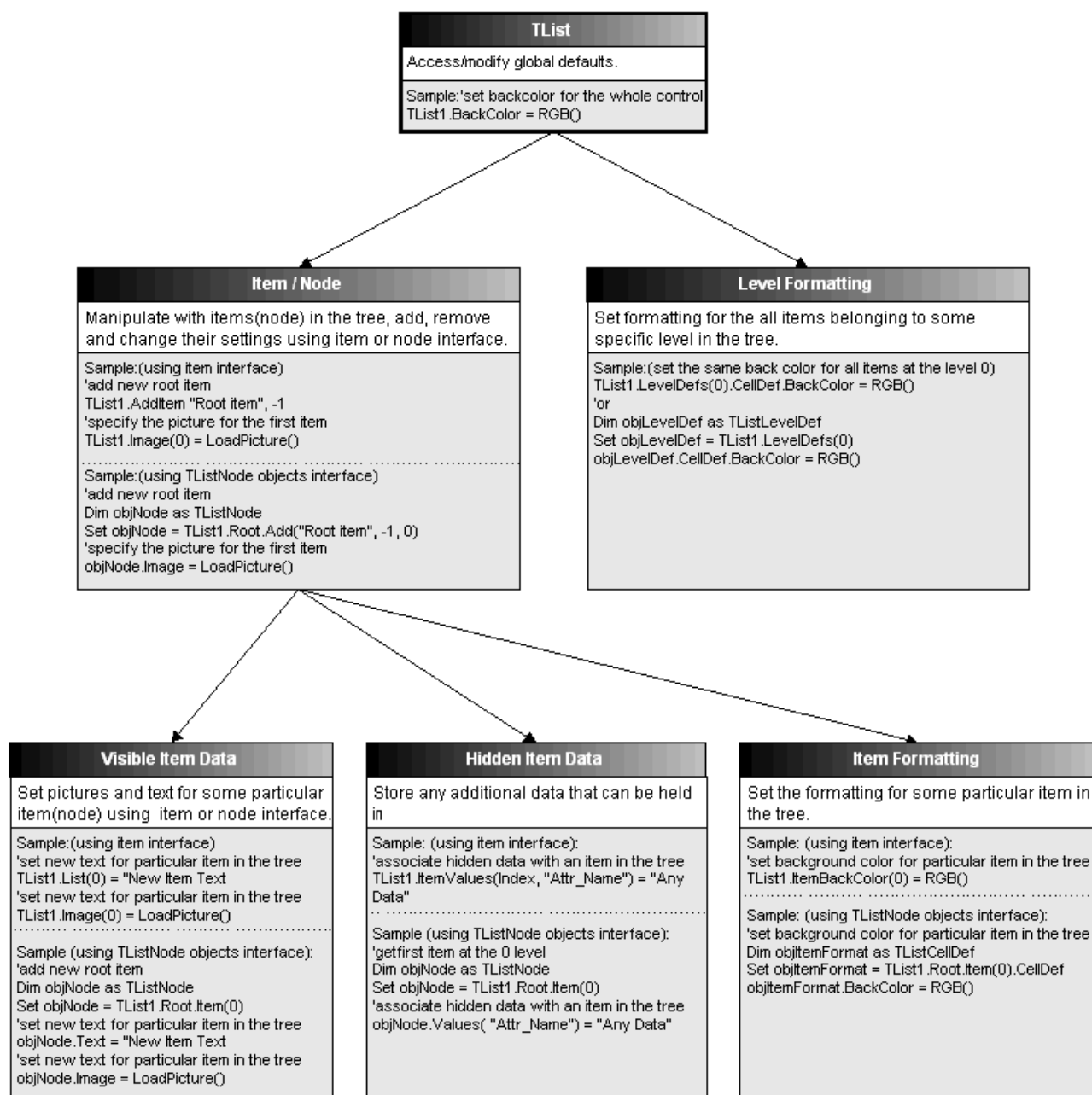


### See also:

Tree Data/Objects Diagram  
Grid Data/Objects Diagram  
Objects/Relationship Diagram

### **Tree Data/Objects Diagram**

The following diagram shows how to access TList data structures holding tree data and formatting.

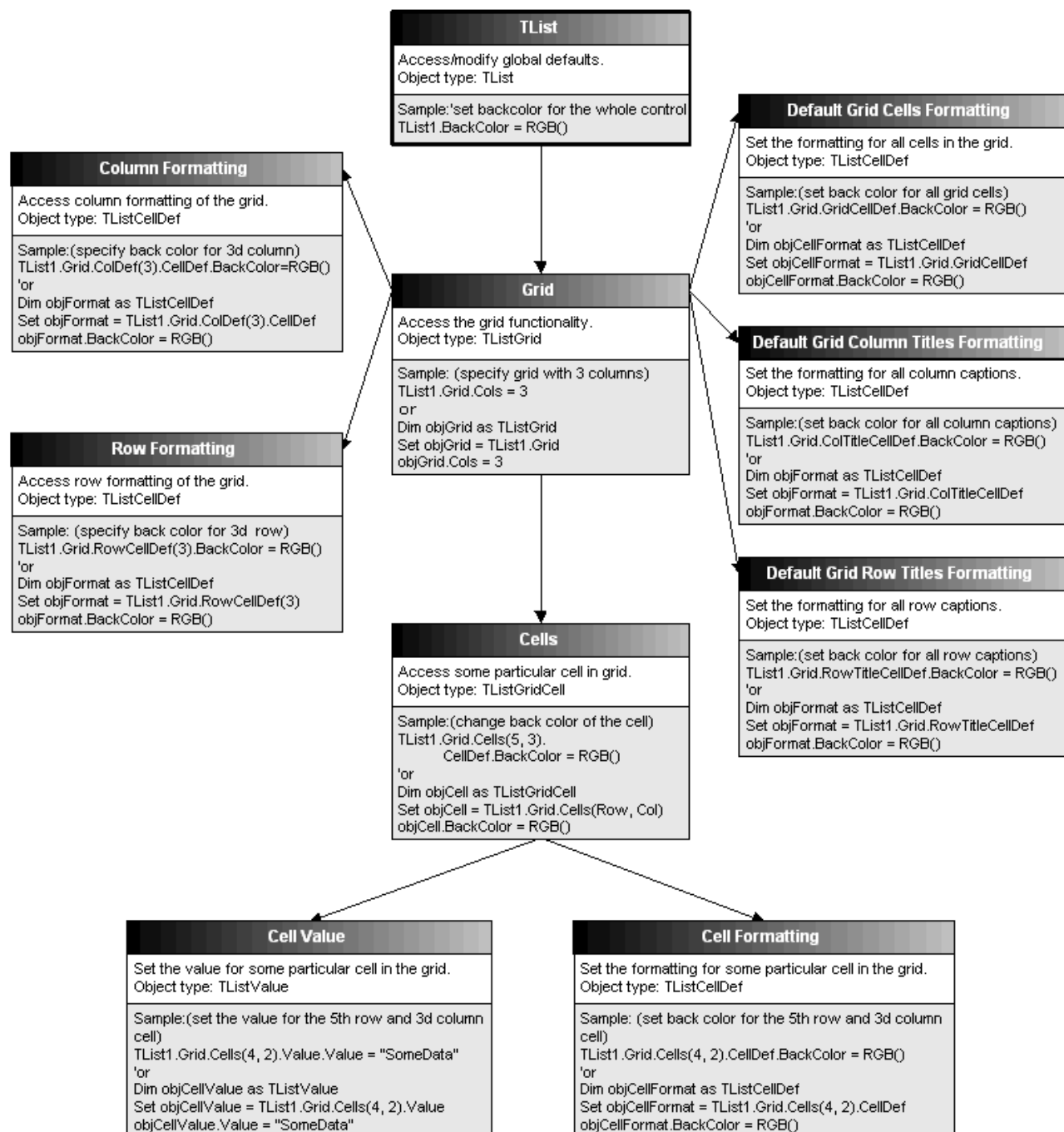


### See also:

Features Diagram  
Grid Data/Objects Diagram  
Objects/Relationship Diagram

## Grid Data/Objects Diagram

This diagram shows how to access TList grid structures holding data and formatting for individual cells and default formatting for rows and columns.



### See also:

Features Diagram

Tree Data/Objects Diagram

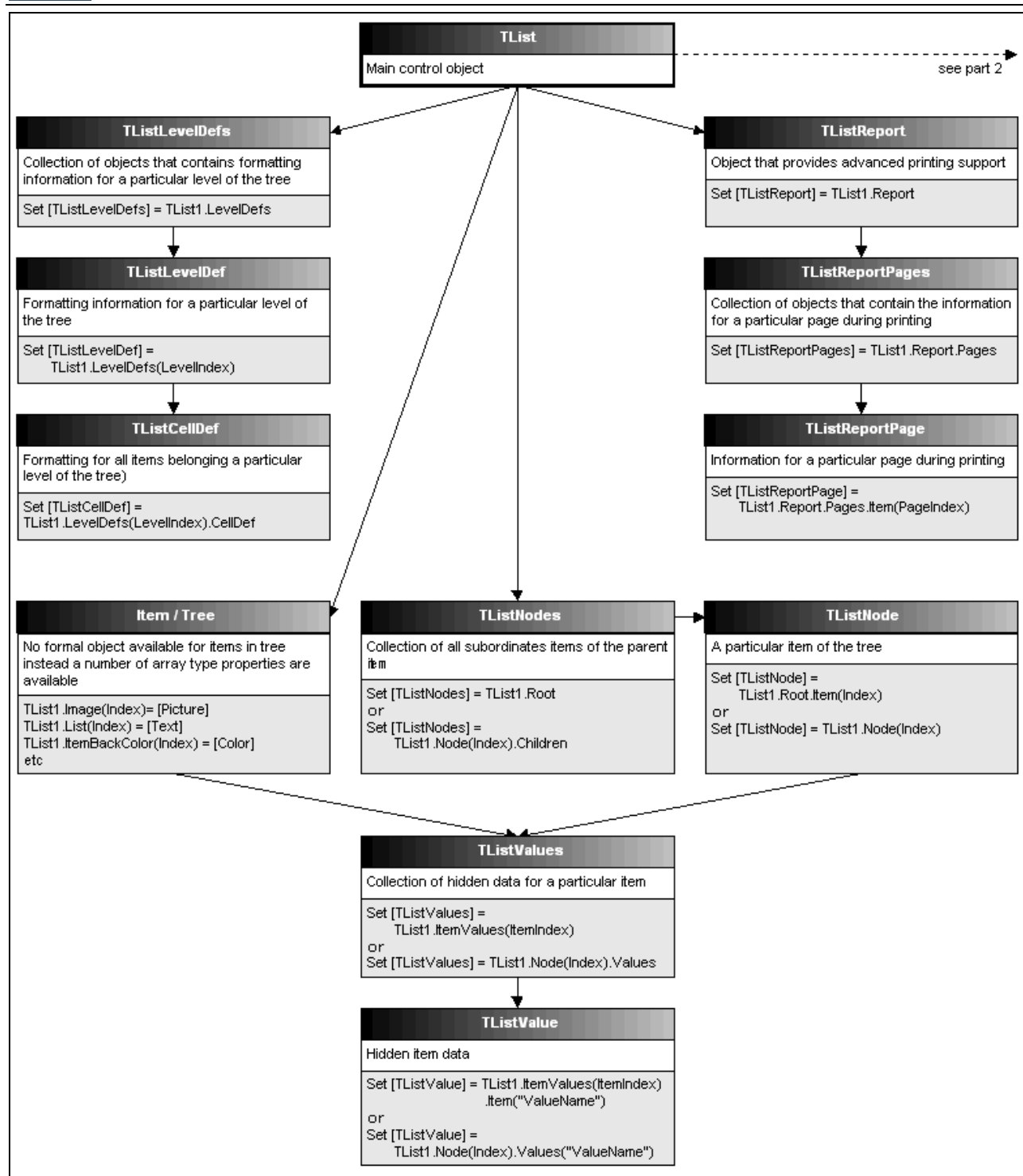
Objects/Relationship Diagram



## Objects/Relationship Diagram

These two diagrams show the relationships between objects and collections exposed by TList and the data or structures these objects represent.

### Part 1



```

classDiagram
    class TListColDefs {
        Collection of columns of the grid
        Set [TListColDefs] = TList1.Grid.ColDefs
    }
    class TListColDef {
        A column of the grid
        Set [TListColDef] = TList1.Grid.ColDefs(ColIndex)
    }
    class TListCellDef {
        Formatting for all cells of the column (exclude title cell)
        Set [TListCellDef] = TList1.Grid.ColDefs(ColIndex).CellDef
    }
    class TListCellDef_titles {
        Formatting for all column titles
        Set [TListCellDef] = TList1.Grid.ColTitlesCellDef
    }
    class TListCellDef_grid {
        Formatting for all grid cells (excluding column and row titles)
        Set [TListCellDef] = TList1.Grid.GridCellDef
    }
    class TListCellDef_row {
        Formatting for all row titles
        Set [TListCellDef] = TList1.Grid.RowTitlesCellDef
    }
    class TListGrid {
        Grid object
        Set [TListGrid] = TList1.Grid or TList1.ItemGrid(Index)
    }
    class TListGridCell {
        Particular cell of the grid
        Set [TListGridCell] = TList1.Grid.Cells(Row, Col)
    }
    class TListValue {
        Value of a particular cell of the grid
        Set [TListValue] = TList1.Grid.Cells(Row, Col).Value
    }
    class TListCellDef_cell {
        Formatting for a particular cell of the grid
        Set [TListCellDef] = TList1.Grid.Cells(Row, Col).CellDef
    }

    TListColDefs --> TListColDef
    TListColDef --> TListCellDef
    TListGrid --> TListColDefs
    TListGrid --> TListCellDef_titles
    TListGrid --> TListCellDef_grid
    TListGrid --> TListCellDef_row
    TListGrid --> TListCellDef_cell
    TListGrid --> TListGridCell
    TListGridCell --> TListValue
    TListGridCell --> TListCellDef_cell
    
```

The diagram illustrates the relationships between various classes in the TListGrid framework. The central class is **TListGrid**, which represents the grid object. It has several associated classes:

- TListColDefs**: A collection of columns of the grid. It is associated with **TListColDef** (A column of the grid).
- TListCellDef**: Formatting for all cells of the column (exclude title cell). It is associated with **TListGrid** and **TListCellDef** (Formatting for all column titles).
- TListCellDef** (Formatting for all column titles): Formatting for all column titles. It is associated with **TListGrid**.
- TListCellDef** (Formatting for all grid cells (excluding column and row titles)): Formatting for all grid cells (excluding column and row titles). It is associated with **TListGrid**.
- TListCellDef** (Formatting for all row titles): Formatting for all row titles. It is associated with **TListGrid**.
- TListCellDef** (Formatting for a particular cell of the grid): Formatting for a particular cell of the grid. It is associated with **TListGrid** and **TListGridCell**.
- TListGridCell**: Particular cell of the grid. It is associated with **TListGrid** and **TListValue** (Value of a particular cell of the grid).
- TListValue**: Value of a particular cell of the grid. It is associated with **TListGridCell**.

The diagram also shows the following relationships:

- TListColDefs** is associated with **TListColDef**.
- TListColDef** is associated with **TListCellDef**.
- TListGrid** is associated with **TListColDefs**.
- TListGrid** is associated with **TListCellDef** (Formatting for all column titles).
- TListGrid** is associated with **TListCellDef** (Formatting for all grid cells (excluding column and row titles)).
- TListGrid** is associated with **TListCellDef** (Formatting for all row titles).
- TListGrid** is associated with **TListCellDef** (Formatting for a particular cell of the grid).
- TListGrid** is associated with **TListGridCell**.
- TListGridCell** is associated with **TListValue**.
- TListGridCell** is associated with **TListCellDef** (Formatting for a particular cell of the grid).

Features Diagram  
Tree Data/Objects Diagram  
Grid Data/Objects Diagram

---

## TListView features and programming techniques

**TListView** is a drop-in replacement for Microsoft TreeView control. It provides support for standard MS TreeView properties and presentation functionality and features. In addition TListView also provides a TList interface via special "TList" object property. All TList events are duplicated inside TListView control (using TListXXX prefix) and are generated before standard MS TreeView events. If desired the TList events may be turned off by setting FireTListEvents property.

### See also:

MSTreeView and TListView compatibility

TListView and TList

## MSTreeView and TListView compatibility

**TListView** control is compatible with MS TreeView at the property level. So it is possible to replace **MS TreeView** with **TListView** inside most VB applications without changing source code.

---

**Note:** The TList installation kit includes a special project converter application to simplify the process of replacing the MS TreeView controls in an existing project. This converter application automatically replaces MS TreeView with instances of TListView controls in a VB project.

---

### Properties/Events compatibility

**TListView** control provides support for almost all of MS TreeView control properties and events. Here is a list of all unsupported or functionally limited MS TreeView properties and events:

- **HideSelection** property is not supported. Always returns FALSE.
- **LineStyle** property supports only tvwRootLines constant. The value tvwTreeLines is not supported by TListView.
- **DropHighlight** property can not be set. Read-only.
- **VisibleCount** property returns the number of currently visible items in a control window. This number can change depending on which items are visible at the time the property is read.
- **CreateDragImage** property (**Node** object) are not supported in this version of **TListView** control.

### ImageList control compatibility

**TListView** control works with the MS ImageList control in almost the same manner as MS TreeView does. There is only one difference, after adding or removing an image to/from ImageList control at run-time the **TListView** method UpdateImages must be called in order to notify TListView control about these changes.

### Interfaces compatibility

**TListView** control provides exact replacements for all MS TreeView OLE interfaces (**Node**, **Nodes**, **DataObject** and **DataObjectFiles**).

### Windows Messages

**TListView** does not support any Windows Messages which may otherwise be used for control of **MS TreeView**.

### See also:

TListView features and programming techniques  
TListView and TList

## TListView and TList

**TListView** control provides significant additional functionality beyond MS TreeView. Since **TListView** control is based on TList control and uses it internally it is possible and helpful to use advanced features which TList control provides.

TListView provides the standard properties of the Microsoft MS TreeView control.

These are not documented here – for further information refer to Microsoft Documentation on the MS TreeView control.

The additional properties listed below provide access to the TList interface from within **TListView** control:

Property/method	Description
<b>TList</b> ( and from there all direct properties of TList )	This property returns the reference to the TList object that is used by TListView control internally. Using the TList object an application has access to the underlying TList support (note: TList virtual functionality is not accessible from within TListView control). Ex: TListView.TList.ItemFontName = "Arial"
<b>UpdateImages</b>	This method must be called in order to notify TListView control about changes to any pictures inside an associated ImageList control
<b>FireTListEvents</b>	This property disables/enables firing TList events. TListView events designed to replicate MSTreeView events will still be fired.
<b>Redraw</b>	This property controls repainting of the control.
<b>SaveData</b>	Saves TListView items and property settings in specified .TLV data file.
<b>LoadData</b>	Loads items and properties settings from a .TLV data file. This file can be saved via <b>SaveData</b> TList method.
<b>TLNode</b>	This property returns a reference to corresponding TListNode object.
<b>NodeFromGridCell, NodeFromItem, NodeFromTListNode</b>	These methods return a TListView node object, or an empty object (NULL) if there is no corresponding TreeView node, for instance if GridCell is TreeGrid Column Title cell. These methods simplify converting a TList ItemIndex, TListNode or TListGridCell reference to a TreeView Node
<b>SelectionMode</b>	This property determines the selection mode of the TListView component and supports single selection or one of several multi-selection modes (the same selection modes as are supported using the MultiSelect property of TList).
<b>SelectedNodes</b>	This property returns a TVwSelectedNodes collection containing all selected nodes in TListView.

**Note:** When manipulating data in TListTreeView using TList methods, all TListTreeView relative information will be transferred as well. So it is possible to continue using TListTreeView interface to access items after performing the actions via TList interface.  
**There is one limitation:** For compatibility with MS TreeView any duplicates will be eliminated when pasting or loading items with a key already held by another TListTreeView item.

**Note:** To access TList interfaces and constants from within VB code it is necessary to add the TList component as well as the TListTreeView component to the project (it is not necessary to put it on the form just to have it in the toolbox).

Here is a sample illustrating how to add automatic drag/drop functionality within the TLTreeView control using TList interface syntax:

```
Private Sub Form_Load()
'add items to TListTreeView control
Call AddItems(TL_TreeView1) ' some routine using standard TreeView syntax to build list
'turn on automatic drag/drop
TL_TreeView1.TList.AutoDragMode = TLAUTODRAGMODE_ASCHILD
TL_TreeView1.TList.DragIconStyle = TLDRAGICONSTYLE_SMART
End Sub

' the TList OnDragDrop and OnDragOver methods should be called from within the corresponding events
Private Sub TreeView1_DragDrop(Source As Control, x As Single, y As Single)
    TL_TreeView1.TList.OnDragDrop x, y
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, state As Integer)
    TL_TreeView1.TList.OnDragOver x, y, state
End Sub
```

Here is a sample illustrating how to turn on sorting functionality:

```
'getting reference to TListNode object
Dim TmpNode As TListNode
Set TmpNode = TListTreeView1.Nodes("SomeItem").TLNode
'using TListNode object interface for sorting all subordinate items
TmpNode.SortingStyle = TL_B_SORT_IGNORE_CASE
TmpNode.SortingMethod = 0 'sort in the ascending order
```

TListTreeView provides the standard events of the Microsoft MS TreeView control. These are not documented here – for further information refer to Microsoft Documentation on the MS TreeView control.

The following TList specific events may be accessed by users of the **TListTreeView** control (Note: All events starting with TListXXXX prefix are generated only if FireTListEvents property is set to True.):

Event	Description
TListAutoDragRequest	Triggered at the start of drag/drop operation in AutoDragDrop mode.
TListAutoDragComplete	Triggered upon completion of any drag/drop operation in AutoDragDrop mode.
TListAfterEditing	Occurs after item text editing.
TListBeginPage	Generated for each page as it is ready to be printed.
TListCollapse	Occurs when an item is collapsed.

TListDragDropEx	Occurs immediately before OleDragDrop event when using automatic drag/drop support.
TListDragOverEx	Occurs immediately before OleDragOver event when using automatic drag/drop support.
TListEditingKeyDown	Occurs when the user presses a key while in text editing mode.
TListEditingKeyPress	Occurs when the user presses a key while in text editing mode.
TListEditingKeyUp	Occurs when the user releases a key while in text editing mode.
TListEndPage	Generated for each page after TList's data has been printed
TListExpand	Occurs when an item is expanded.
TListGridCellActivate	Occurs right after a cell becomes active (having focus), but before any changes are displayed on the screen.
TListGridCellDeactivate	Occurs when a cell is being deactivated (losing focus), but before the corresponding changes become visible on the screen.
TListGridCellClick	Occurs when the user selects a cell in a grid by clicking the mouse button.
TListGridCellDbClick	Occurs when the user double-clicks a cell in a grid.
TListGridCellRequestEditing	Occurs when the <i>TListGridObject.CellEdit</i> property is set to TLCELLEDIT_BEGIN.
TListGridCellAfterEdititng	Occurs when the <i>TListGridObject.CellEdit</i> property is set to TLEDITTEXT_END or when editing is canceled by the user.
TListGridCellEditingKeyDown	Occurs as a result of a Keyboard actions during editing.
TListGridCellEditingKeyUp	Occurs as a result of a Keyboard actions during editing.
TListGridCellEditingKeyPress	Occurs as a result of a Keyboard actions during editing.
TListGridRowActivate	Occurs right after a row becomes active (having focus), but before any changes are displayed on the screen.
TListGridRowDeactivate	Occurs when a row is being deactivated (losing focus), but before the corresponding changes become visible on the screen.
TListGridRowTitleClick	Occurs when the user clicks a cell in a title row in any Grid object in TList.
TListGridColumnTitleClick	Occurs when the user clicks a cell in a column in any Grid object in TList.
TListGridCornerTitleClick	Occurs when the user clicks a corner cell in any Grid object in TList.
TListItemActivate	Occurs after an item becomes active (gets the focus).
TListItemDeactivate	Occurs right before an item becomes inactive (loses the focus).
TListItemClick	Occurs when the user selects an item in a list

	box by clicking the mouse button.
TListItemDbClick	Occurs when the user double-clicks a cell item in a grid.
TListMarkClick	Occurs when the mark picture associated with an item is clicked.
TListMarkDbClick	Occurs when the mark picture associated with an item is double-clicked.
TListMouseWheel	Occurs when user rotates mouse wheel.
TListOLEDragDrop	Occurs when an OLE object is dropped into the control.
TListOLEDragOver	Occurs when an OLE object is dragged over the control.
TListOLECompleteDrag	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.
TListOLEGiveFeedback	Occurs after every OLEDragOver event and allows the source TList component to provide visual feedback to the user, such as changing the mouse cursor
TListOLESetData	Occurs on a source component when a target component performs the GetData method on the source's TListDataObject object, but before the data for the specified format has been loaded.
TListPictureClick	Occurs when the picture associated with an item is clicked.
TListOLEStartDrag	Occurs when a component's OLEDrag method is called (explicitly or not).
TListPictureDbClick	Occurs when the picture associated with an item is double-clicked.
TListPlusMinusClick	Occurs when the plus/minus picture associated with an item is clicked.
TListPlusMinusDbClick	Occurs when the plus/minus picture associated with an item is double-clicked.
TListPreparePage	Generated for each page after the PrepareForPrinting method has been called
TListTitlesResize	Generated when end-user starts to change a column width or row height by dragging a grid line.
TListRequestEditing	Occurs after the ItemEditText property has been set, but before editing begins.
TListHScroll	Occurs when the user scrolls the control horizontally.
TListVScroll	Occurs when the user scrolls the control vertically.

### See also:

TListTreeView features and programming techniques

MSTreeView and TListTreeView compatibility

---

## Using the TDesigner application

TDesigner was developed as a tool to facilitate setting of TList properties and populating TList with list items at design time. Using TDesigner, a complete Tree can be built, formatted, and saved in a data file for later use, without writing a single line of code. Loading the pre-built list in an application is considerably faster than building a list at run-time. The data file can also be passed to TList on a Web site allowing the design of a Web page using TList without need for any programming.

TDesigner is a 32-bit application, TDESIGN8.EXE, compatible with Windows 95, Windows 98, Windows ME, Windows NT 4, Windows 2000, Windows XP and Windows Vista. Running TDesigner requires the following files: TLIST8.OCX, MSVCRT.DLL, MFC42.DLL, COMCAT.DLL, COMDLG32.DLL, TABCTL32.OCX, THREED32.OCX, MSVBVM60.DLL.

TDesigner has 2 modes of functioning: stand-alone mode, when it is running as a regular application, and design-time mode, when this application is called to edit properties of a TList control placed on a form of either Visual Basic, Visual C, Delphi, or other environment supporting ActiveX Controls.

**Stand-alone mode**, TDesigner can be used to create, view, modify and save TList data files (.TLT files). This includes settings of all item text, grids, associated hidden data, formatting, images, view styles and other aspects of the look and feel of a hierarchic list.

Any .TLT file produced by TDesigner can be later loaded into any TList 8 control at design or run time via **LoadData** method. Also, files created by an application using the **SaveData** TList method can be loaded in TDesigner for viewing and editing.

**Design-time mode**, TDesigner replaces the Property window formerly shown in response to the Property command from control's right mouse menu. In this mode, TDesigner directly sets the properties and characteristics of a TList control placed on a form in the development environment. It is not necessary to create .TLT files when using the control in this way, but TLT files can be loaded at this time to pull in default settings.

---

**License Note:** TDesigner is licensed by Bennet-Tec Information Systems and MAY NOT be shared among individuals or distributed without specific permission by Bennet-Tec. Loading files created by TDesigner on an unlicensed machine into a TList application will cause TList to display an licensing error message to the end-user. Customers intersted in distributing a version of TDesigner with their applications should contact Bennet-Tec Information Systems to discuss licensing options.

---

### **TDesigner Features List**




<b>TDesigner Layout</b>
Design-Time Mode buttons
Using TDesigner Windows
Using Clipboard
Using Drag/Drop
Printing
Hints for Web Site Designers
Setting up ToolTips
Selecting Colors
Setting up Pictures
Modifying Tree Line Settings

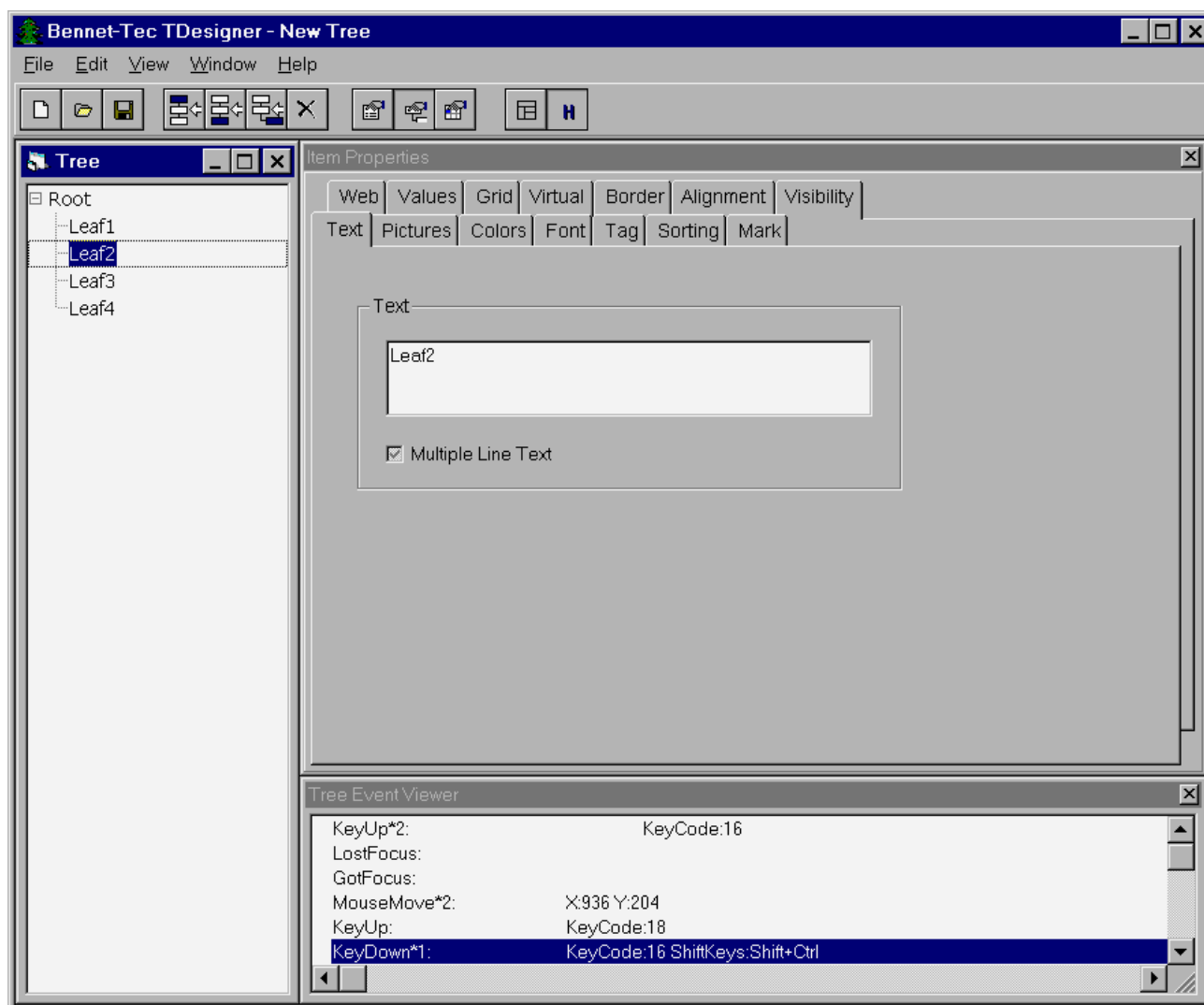


Specifying Drag Drop Settings
Controlling Selection
Controlling Expanding/Collapsing
Controlling Text Display
Controlling Fonts
Controlling Marks
Associating Additional Data with an Item
Specifying Sorting Method
Controlling Miscellaneous Settings
Controlling Item Cell Default Settings
Setting up Background
Specifying Scrollbar Appearance
Setting up Item And Grid Cell Borders
Setting up Item And Grid Cell Alignment
Specifying Virtual Items
Setting up Item Visibility
Setting up Item And Grid Cell Tag
Setting up LevelDefs
Specifying a Tree Grid
Specifying Item Grids
Properties You Cannot Set with TDesigner


## TDesigner Layout

There are five main windows in TDesigner: *Tree*, *Event Viewer*, *Properties*, *Grid Cell Properties* and *Item Properties*.

- In the *Tree* window users may view and edit the contents of a TList control. There user can add or remove any item or edit its properties. Use the *Window\Tree* menu command to display this window.
- In the *Event Viewer* window users can see a log of all TList events that have occurred. Use the *View\Event Viewer* menu command to display this window.
- In the *Properties* window users may edit property settings for the whole control, such as:  
background color, default font etc. Use the  toolbar button or *Window\Properties* menu command to display this window.
- In the *Item Properties* window users may edit settings of each list item. Use the  toolbar button or *Window\Item Properties* menu command to display this window.
- In the *Grid Cell Properties* window users may edit settings of each cell of each grid. Use the  toolbar button or *Window\Grid Cell Properties* menu command to display this window.  
The picture below shows TDesigner with the *Tree*, *Properties* and *ItemProperties* windows open.



## Window Arrangement

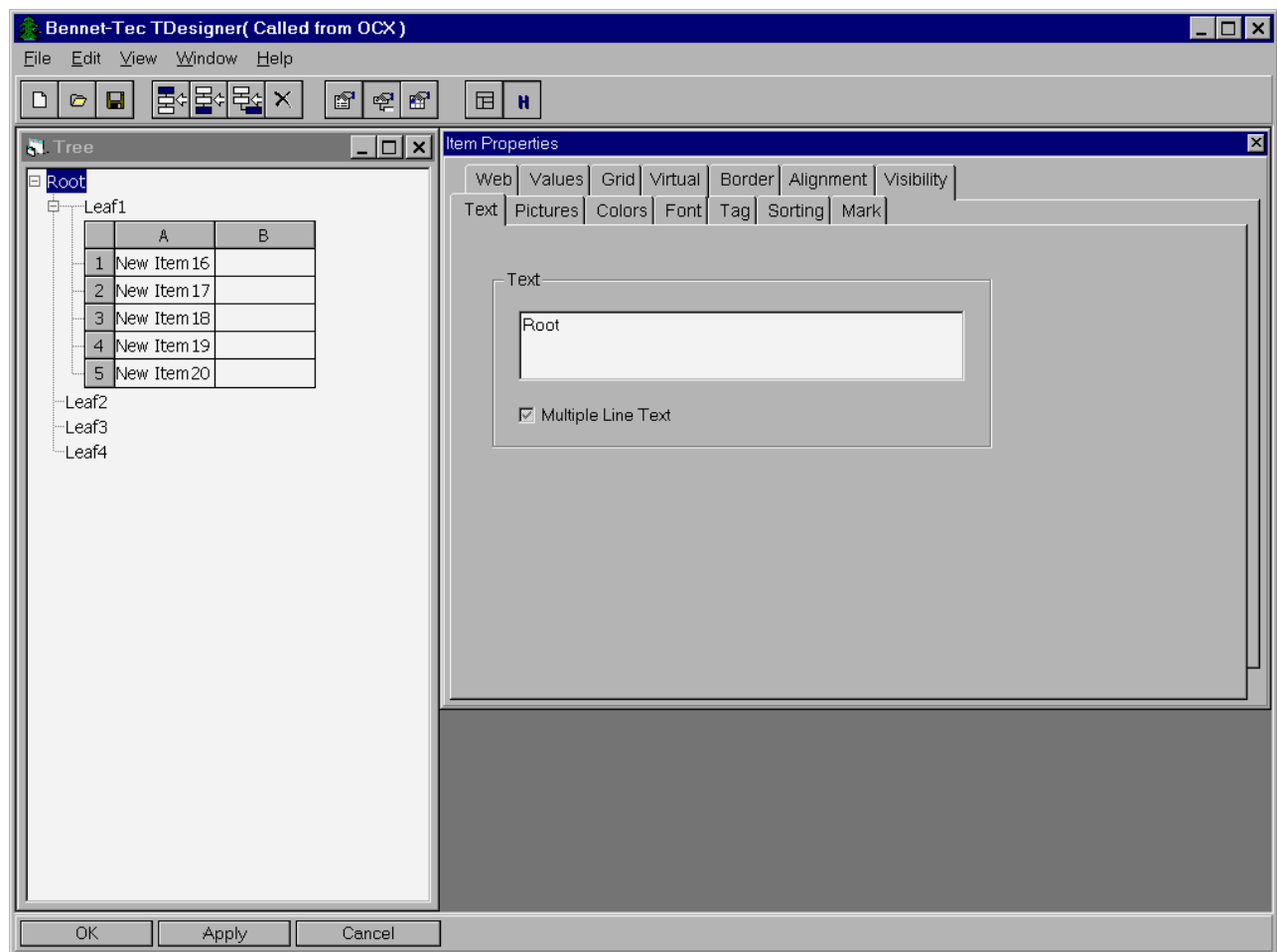
For convenient positioning of *Tree*, *Event viewer*, *Properties*, *Grid Cell Properties*, or *Item Properties* windows, use the  toolbar command or Window\Arrange menu command to change the size and position of these windows.

### See also:

TDesigner Features List

## Design-Time Mode buttons

Three additional buttons, *OK*, *Apply* and *Cancel*, are presented at the bottom of the main window of TDesigner when it is used as a property access tool for TList within a design time software development environment (such as Visual Basic). When TDesigner is called in this mode, it appears as is shown below:




These extra buttons shown at the bottom of the main window of the TDesigner application determine how TDesigner updates information in the TList control whose settings are being modified.

Upon Clicking the *OK* button, TDesigner updates the information in the control and shuts down.

Upon Clicking the *Apply* button, TDesigner updates the information in the control and continues running.




Upon Clicking the *Cancel* button, TDesigner cancels any changes which might have been made since the apply button was last clicked or since the TDesigner application was invokled, and shuts down.

## Displaying Hidden Items

Use the  toolbar button to display items, which are not normally shown. Pressing this button sets the **ShowHiddenItems** property to True.

## Operations

All operations which are listed below can be applied to the currently selected item.

Toolbar button	Menu Command	Operation
	Edit\Insert Item Before	Insert item before selected one
	Edit\Insert Item After	Insert item after selected one
	Edit\Insert As Subordinate	Insert item as subordinate one

	Edit/Delete	Remove selected item
---	-------------	----------------------

## Using TDesigner Windows

### Using the Tree Window

The *Tree* window displays and edits TList contents. In this window, the user can add and remove any item.

### Using the Tree Event Viewer Window

The *Tree Event Viewer* window logs all the TList events which might be fired as a result of user actions (clicking, moving the mouse, etc).. This window may be useful to learn when and in what order TList events occur.

### Using the Properties Window

This window allows modification of almost all properties affecting the overall look and behavior of the TList control.

Property pages (tabs) are used to organize the large number of TList properties which may be set:

- The *Web* page collects all Internet-oriented settings.
- The *ToolTips* page has all settings required to control display of ToolTips.
- The *Colors* page collects properties specifying various colors.
- The *Pictures* page collects properties specifying pictures used in the control.
- The *Font* page collects properties specifying the default font used in the control.
- The *Tree Lines* page collects properties controlling painting of Tree Lines.
- The *Drag/Drop* page collects properties controlling drag/drop operations.
- The *Selection* page collects properties controlling how TList selects/deselects items.
- The *Expanding* page collects properties controlling how TList expands/collapses items and keeps expanded state information in an item.
- The *Text* page collects properties controlling how TList displays text.
- The *Marks* page collects properties controlling how TList displays marks.
- The *Miscellaneous* page collects all the other properties.
- The *LevelDefs* page collects **LevelDefs** settings.
- The *Item Cell Defaults* page collects item cell background, border and alignment properties.
- The *Background* page collects properties controlling how the TList background is displayed.
- The *Grid* page collects properties of the Tree Grid object.
- The *Scrollbars* page collects properties controlling scrollbar appearance.

### Using the Item Properties Window

This window shows properties of the item currently selected in the *Tree* window.

Property pages (tabs) are used to help organize and access the list item properties which may be set:

- The *Web* page contains a URL which, if specified, can be used to point to a destination for a hyperlink jump.
- The *Text* page specifies the text shown in the item.
- The *Colors* page specifies the colors with which the item is drawn.
- The *Font* page specifies the text font.

- The *Pictures* page specifies the pictures drawn in the item.
- The *Mark* page specifies the mark picture associated with the item.
- The *Sorting* page specifies the sorting method used for children of the item.
- The *Values* page specifies additional data associated with the item.
- The *Border* page specifies border drawn around the item.
- The *Alignment* page specifies how item text and picture are aligned.
- The *Virtual* page specifies whether the item has virtual children.
- The *Visibility* page specifies whether the item is always hidden.
- The *Tag* page specifies the tag value associated with the item.
- The *Grid* page specifies item grid parameters for the item.

### Using Grid Cell Properties Window

This window shows properties of the grid cell currently selected in the *Tree* window.

Property pages (tabs) are used to help organize and access the list of grid cell properties which may be set:

- The *Value* page specifies the value shown in the grid cell.
- The *Pictures* page specifies the pictures drawn in the grid cell.
- The *Colors* page specifies the colors with which the grid cell is drawn.
- The *Font* page specifies the text font.
- The *Border* page specifies the border style and color of the grid cell.
- The *Alignment* page specifies the text, picture and text / picture alignment settings of the grid cell.
- The *Tag* page specifies the tag value associated with the grid cell.

#### See also:

TDesigner Features List

### Specifying TDesigner Defaults

When TDesigner starts it automatically loads the DEFAULT.TLT file which is in the TDesigner subdirectory of your TList installation directory. You can modify that file so that it sets most TList properties to defaults you prefer. This works like AUTOLOAD.VBP file in Visual Basic.

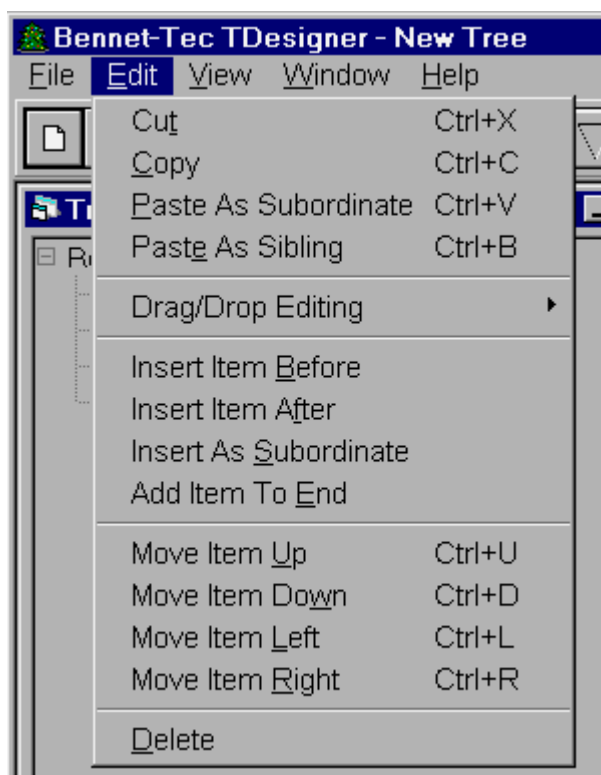
#### See also:

TDesigner Features List

### Using Clipboard

TDesigner supports standard clipboard commands to exchange the tree data between running instances of TDesigner.

To copy/paste data to/from the Windows clipboard use standard Edit\Copy or Edit\Paste menu commands or respective key strokes as shown below:



Note if multi-selection mode is turned on, only selected items will be affected by these menu commands or key strokes.

**See also:**

TDesigner Features List

## Using Drag/Drop

You can use drag/drop to edit the tree. With drag/drop you can move any branch of the tree to a new position or copy any branch into any other location on the tree. (by default drag/drop operations will MOVE items and branches within the tree. To copy depress the CNTRL key while dropping.)

To turn on/off this feature use either following menu commands:



or respective toolbar buttons:



### See also:

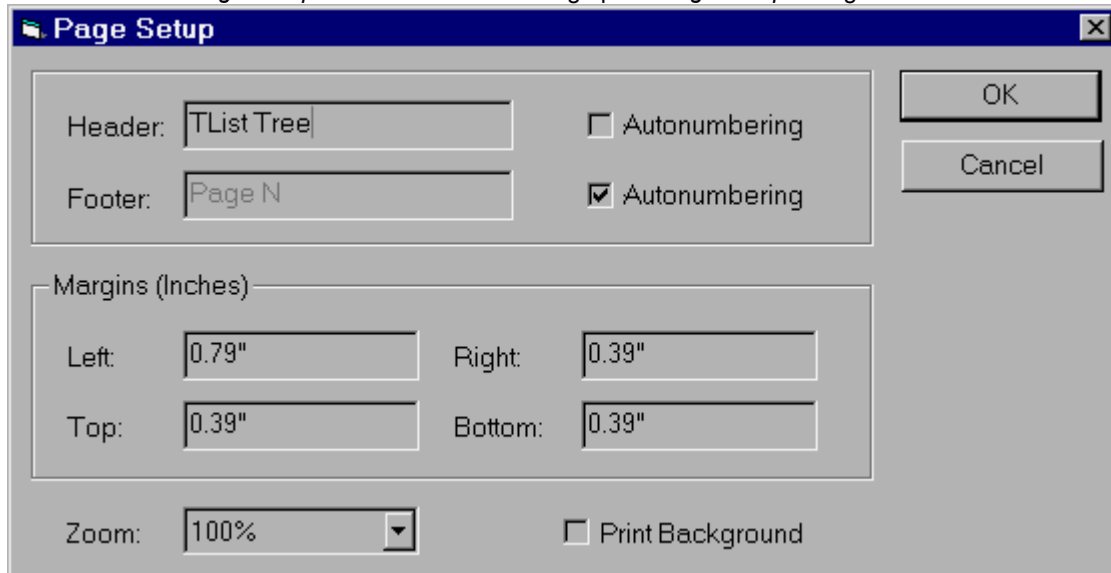
TDesigner Features List

## Printing

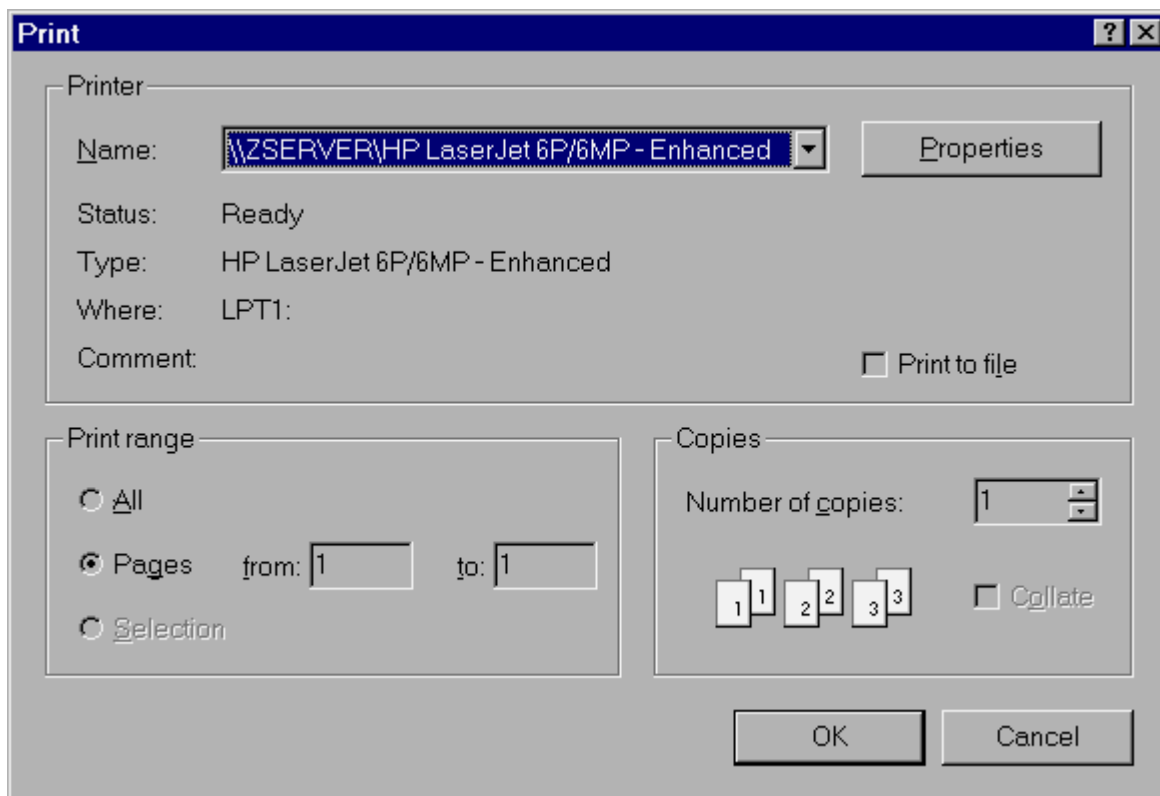
TDesigner6 supports multi page printing.

Use the following procedure to print a tree from TDesigner:

- Create the desired Tree structure within TDesigner or load the desired TLT file from disk.
- Use the *File\Page Setup* menu command to bring up the *Page Setup* dialog which looks as follows:



- Specify margins, the zoom factor, header and footer information as needed
- Use the *File\Print* menu command to actually start printing:



#### See also:

TDesigner Features List

## Hints for Web Site Designers

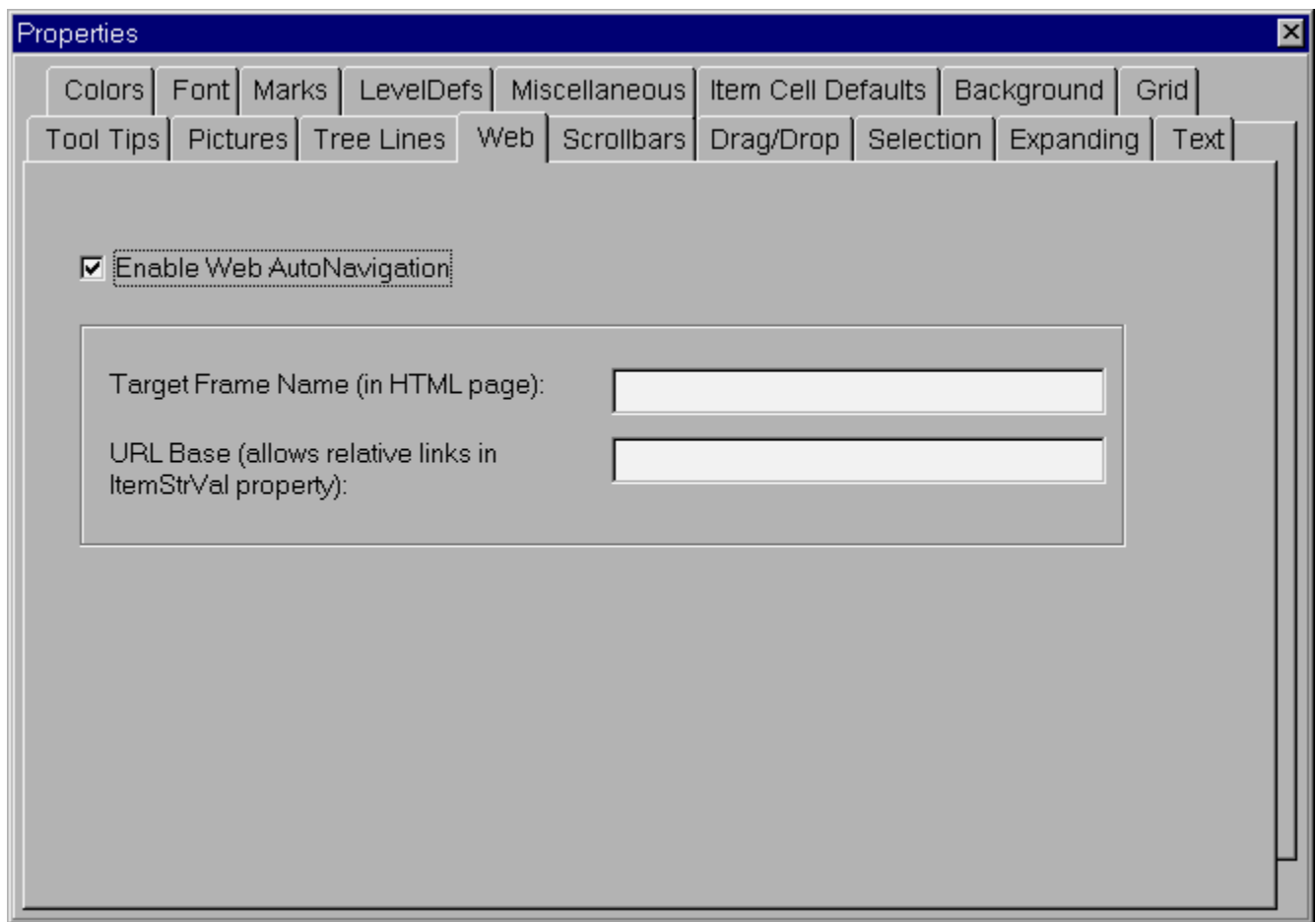
### 2-Frame Scheme

Placing the TList control into one frame on the left side of a web page, and your actual content within a second frame on the right is the most common way of using TList on a Web page.

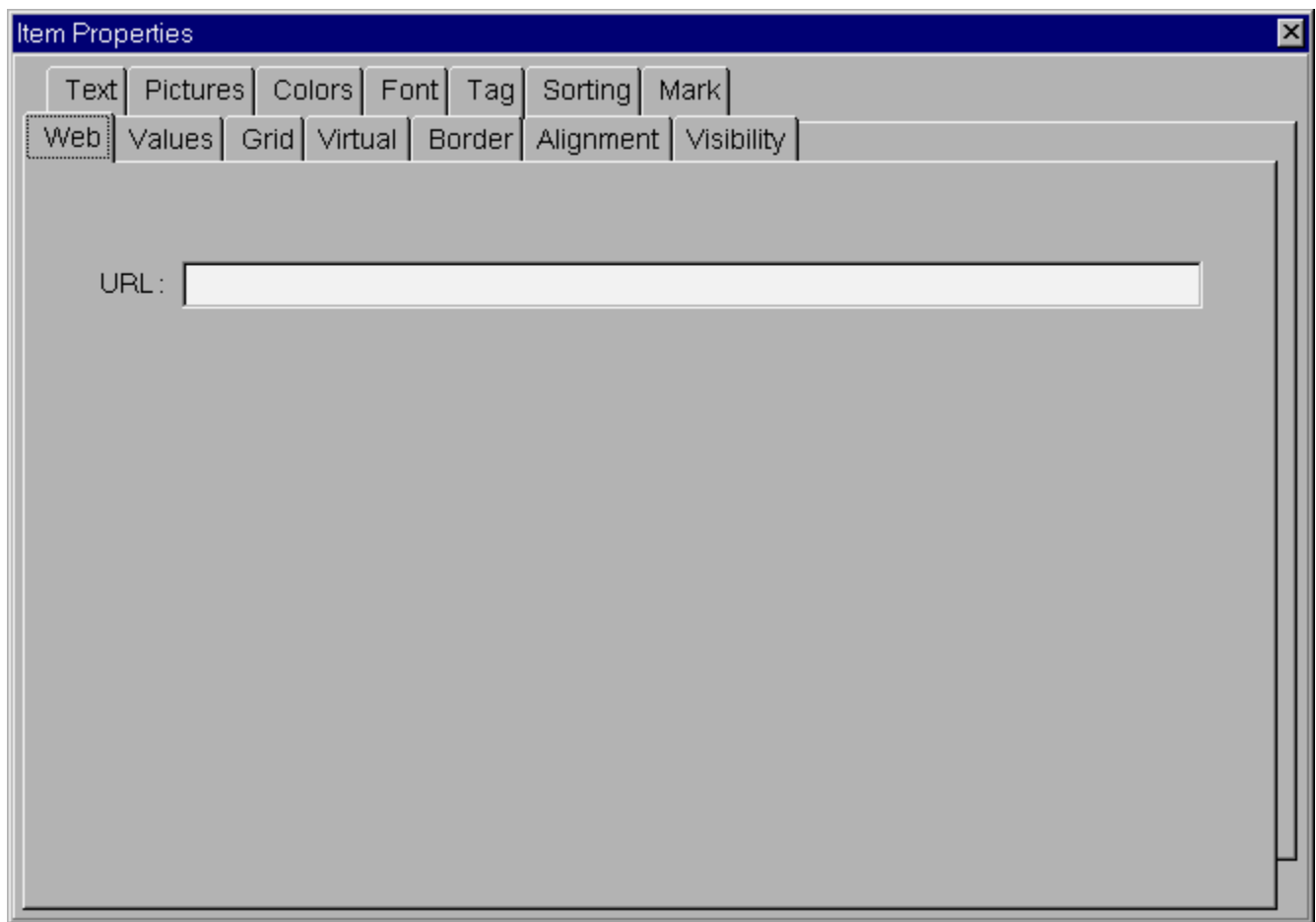
Follow the steps below to create a .TLT data file which sets up TList as a table of contents to your Web site in 2-frame mode without requiring a line of code:

- create the tree you would like to be shown in TList.
- select Web property page from the Properties dialog.





- Check *Enable Web AutoNavigation* checkbox (use **WebAutoNavigate** property to access this functionality at run time).
- Enter the name of the right frame in the *Target Frame Name* text box (use the **WebTargetFrame** property to access this functionality at run time).
- You can specify the URL of the Web server where files are located. This is not necessary. If specified, TList can use this as a Base URL allowing it to work with relative URLs, for example:  
    main.htm, or /Company/Success Story.htm  
(use **WebUrlBase** property to access this functionality at run time).
- In the Tree Window select an item for which you would like to specify a URL for the hyperlink jump.
- select Web property page from the Item Properties dialog.



- Enter the name of the HTML page to refer to. This is the page whose content will be shown in the right frame of the web site. (use the **ItemURL** property to specify a URL at run time).
- repeat 3 steps above for each item which must have a URL.
- EDIT the sample TLDEMO.HTM file in your "*TLIST8\Samples.WWW\Sample 1. The Simplest HTML Page*" directory, replacing the specified .TLT file identified in the DATA = section with a reference to your own .TLT file.
- save the file and try it in Internet Explorer.

### **How to Minimize the Size of Your .TLT File**

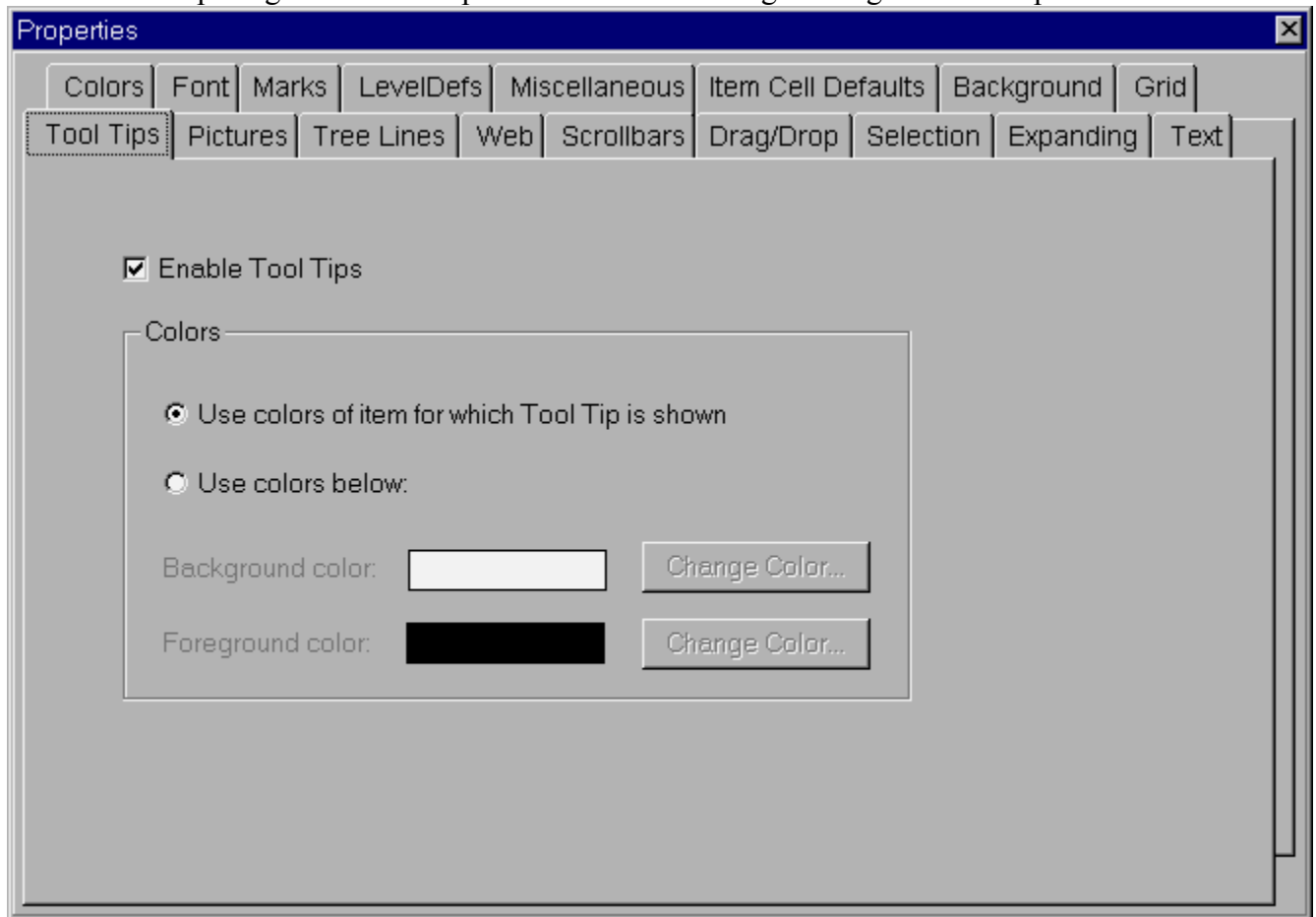
The size of the TList data file is critical if you want to minimize the download time for TList Data. Aside from the number of items in the list, the principle factor determining the size of a TLT file is the number of distinct images stored in the Tree and the size of each image. TList optimizes data storage, saving only one copy of each distinctly loaded picture regardless of the number of items associated with it. Thus assigning the same image to many items does not increase file size. The color resolution of images is very significant. The color resolution of the images saved in a TLT file is determined by the color resolution setting of the PC when the data file is saved. Saving a TLT file on a system set to 16 or 256 colors will greatly decrease the file size as compared to a TLT file saved on a high color resolution setting..

#### **See also:**

TDesigner Features List

## Setting up ToolTips

Use the ToolTips Page from the Properties Window to change settings for ToolTips.



Check the *Enable ToolTips* checkbox to turn on displaying of ToolTips (use **ToolTipsMode** property to do this at run time). The ToolTips Delay box specifies how quickly the ToolTips window should be shown when the mouse is over a item which can not be shown in full within TList or within the specified height/width for an item or grid cell.

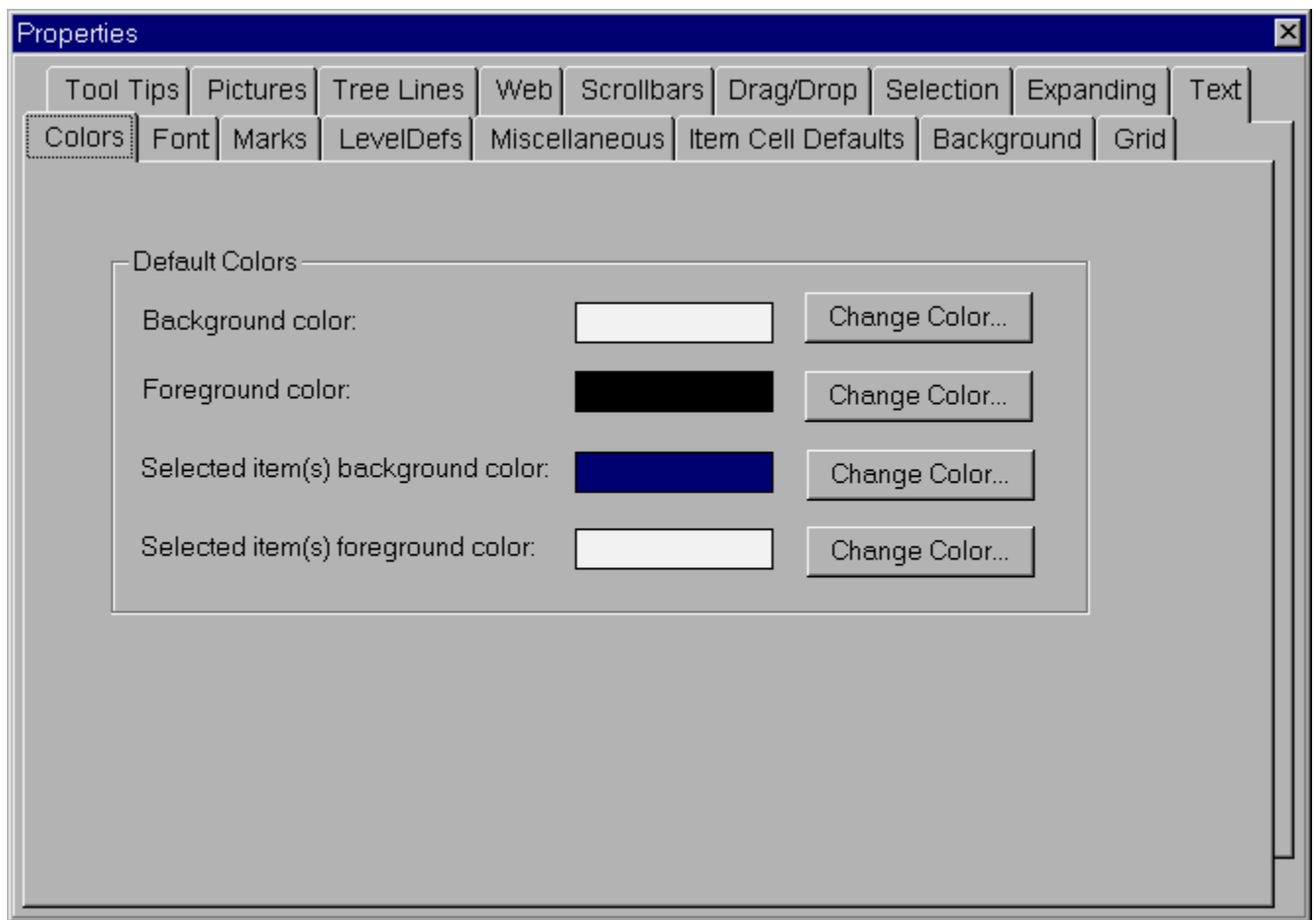
The other controls handle colors of the tool tip box (use **ToolTipsViewStyle**, **ToolTipsBackColor**, and **ToolTipsForeColor** properties to do this at run time).

### See also:

TDesigner Features List

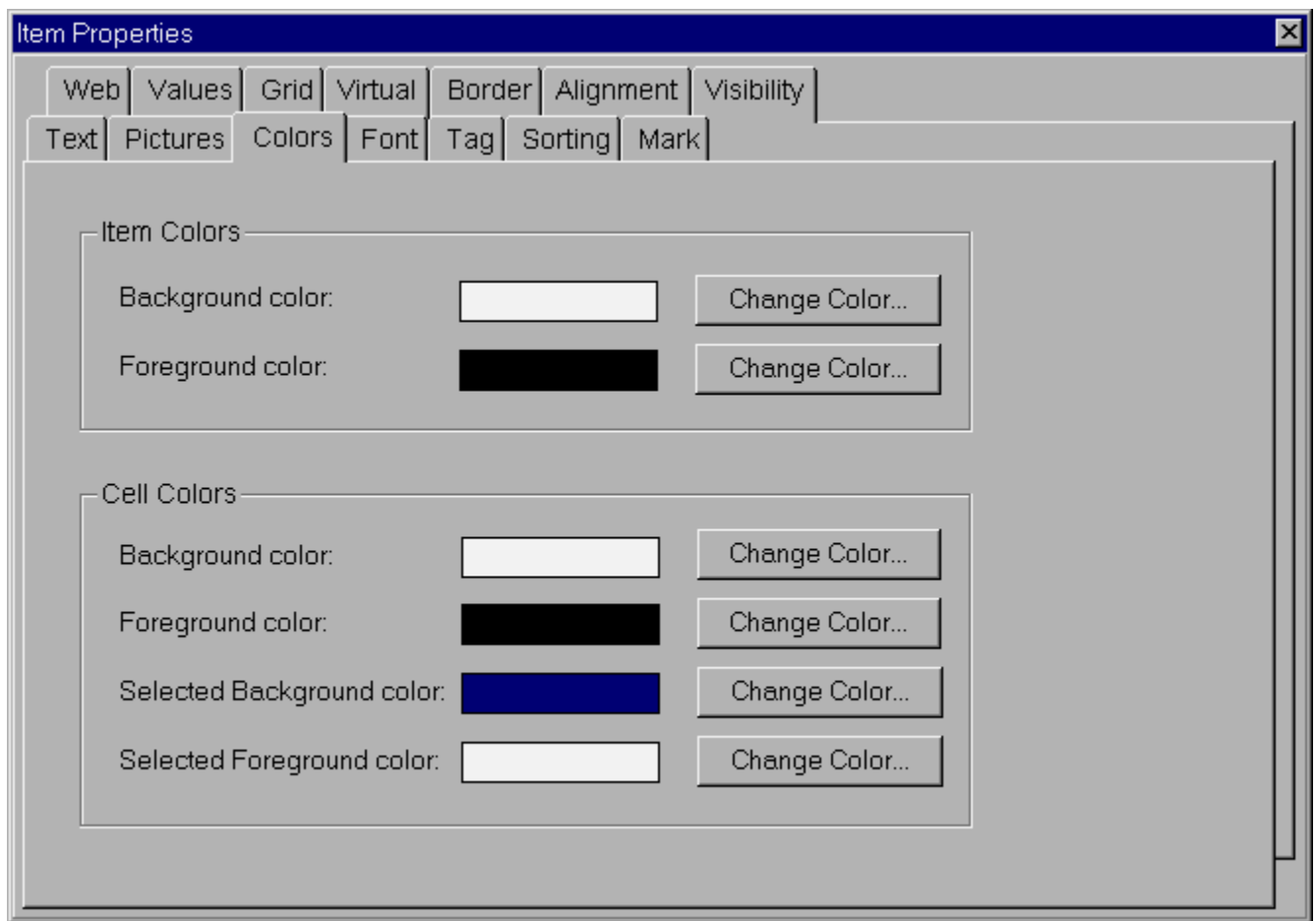
## Selecting Colors

Use the Colors Page from the Properties Window to change settings for the whole TList:



Use **BackColor**, **ForeColor**, **SelBackColor**, **SelForeColor** properties to control colors at run time.

Use the Colors Page from the Item Properties Window to change the colors for currently selected item in the Tree Window item:

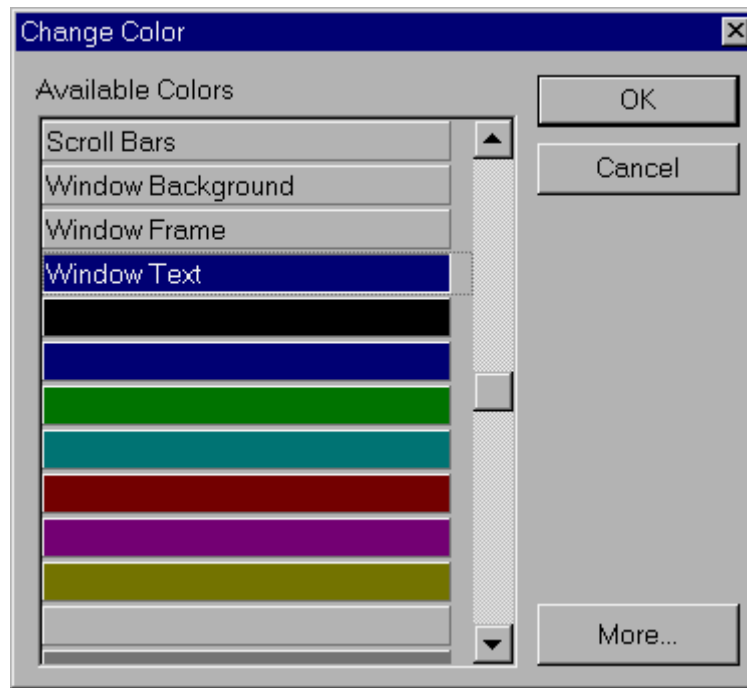


Use **ItemBackColor**, **ItemForeColor**, **ItemCell(ItemIndex&).BackColor**, **ItemCell(ItemIndex&).ForeColor**, **ItemCell(ItemIndex&).SelBackColor** and **ItemCell(ItemIndex&).SelForeColor** properties to control colors at run time.

Use the Colors Page from the Grid Cell Properties Window to change the colors for the currently selected in the Tree Window grid cell.

Use **BackColor**, **ForeColor**, **SelBackColor**, **SelForeColor** properties of Grid Cell's **CellDef** object to control colors at run time.

In response to the *Change Color* button the following dialog appears:



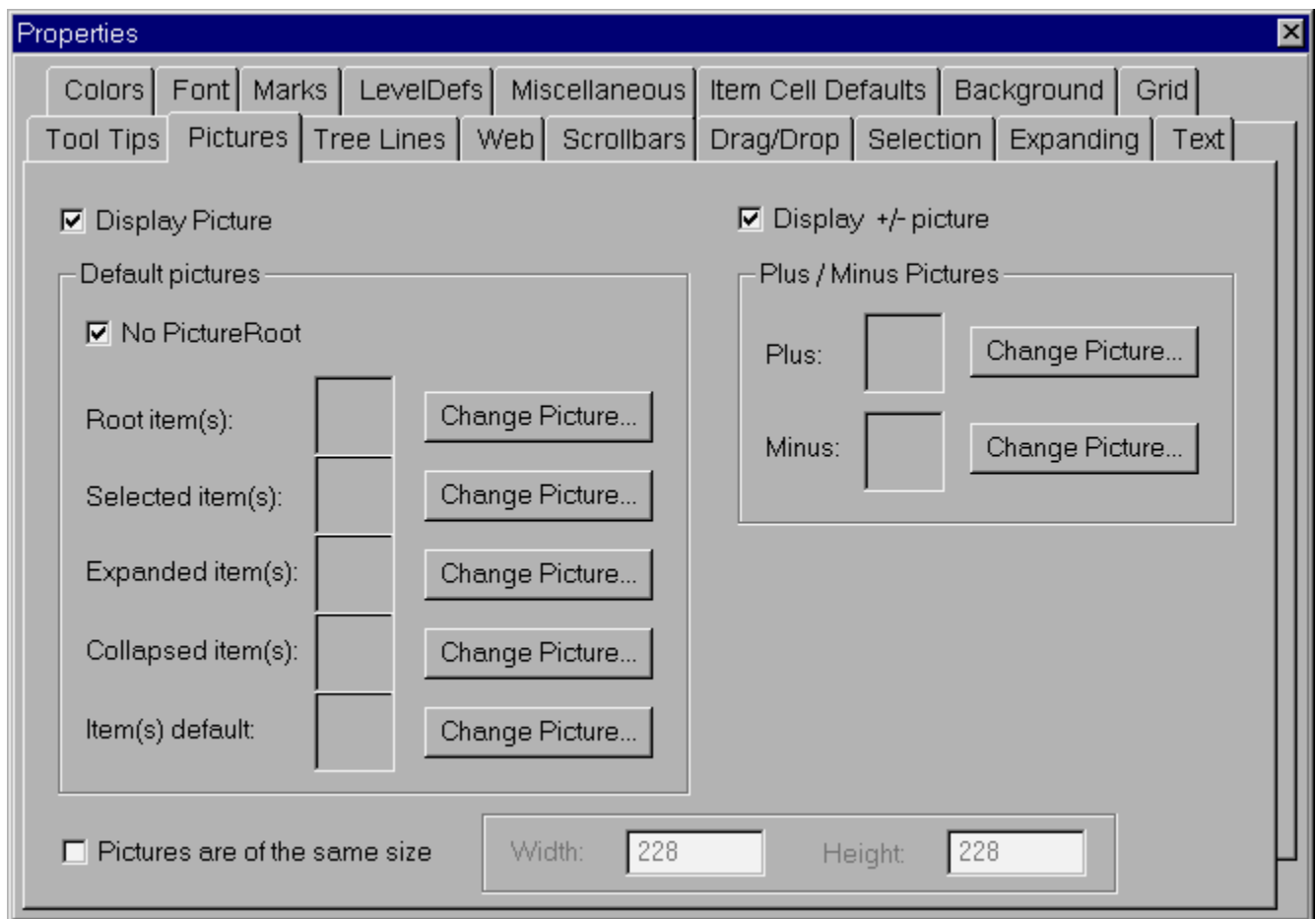
All system colors and 16 most frequently used colors (corresponding to the QBColor function in Visual Basic) are shown in this dialog. Select the one you need and choose *OK*. To select a color which is not on the list, make use of the *More* button.

**See also:**

TDesigner Features List

## Setting up Pictures

Use the Pictures Page from the Properties Window to set default pictures for the whole TList:



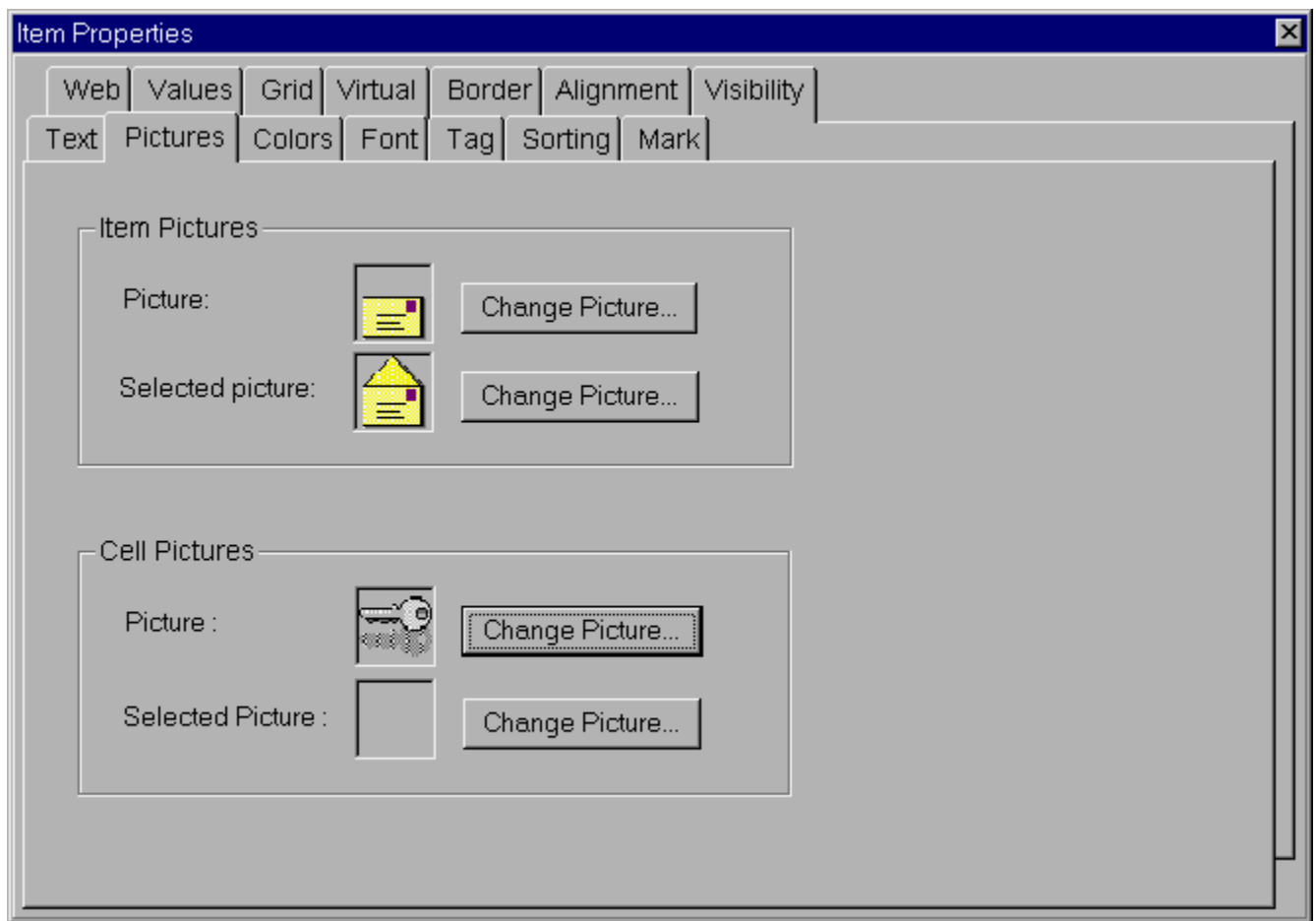
Check the *Display Picture* checkbox to turn on displaying of pictures (use **ViewStyle** property to do this at run time).

Check the *Display + / - Picture* checkbox to turn on displaying of plus / minus pictures (use **ViewStyleEx** property to do this at run time).

Check the *Pictures are of the same size* checkbox to display all the pictures stretched according with height and width specified in textboxes *Height* and *Width* (use **ImageStretch**, **ItemImageDefWidth**, **ItemImageDefHeight** properties to do this at run time).

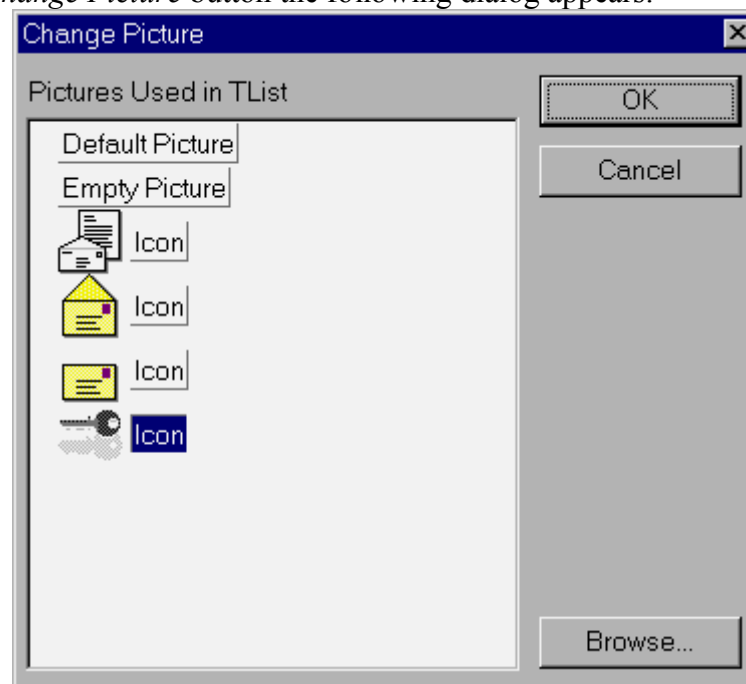
Use **PictureRoot**, **PictureInverted**, **PictureOpen**, **PictureClosed**, **PictureLeaf**, **Image**, **InvImage**, **PicturePlus** and **PictureMinus** properties to control pictures at run time.

Use the Pictures Page from the Item Properties Window to change the picture for the currently selected item in the Tree Window item:



Use the Pictures Page from the Grid Cell Properties Window to change the picture for the currently selected item in the Tree Window grid cell:

In response to any *Change Picture* button the following dialog appears:





All pictures which are currently in use by TList are shown in this dialog. Select the picture you need and choose OK.

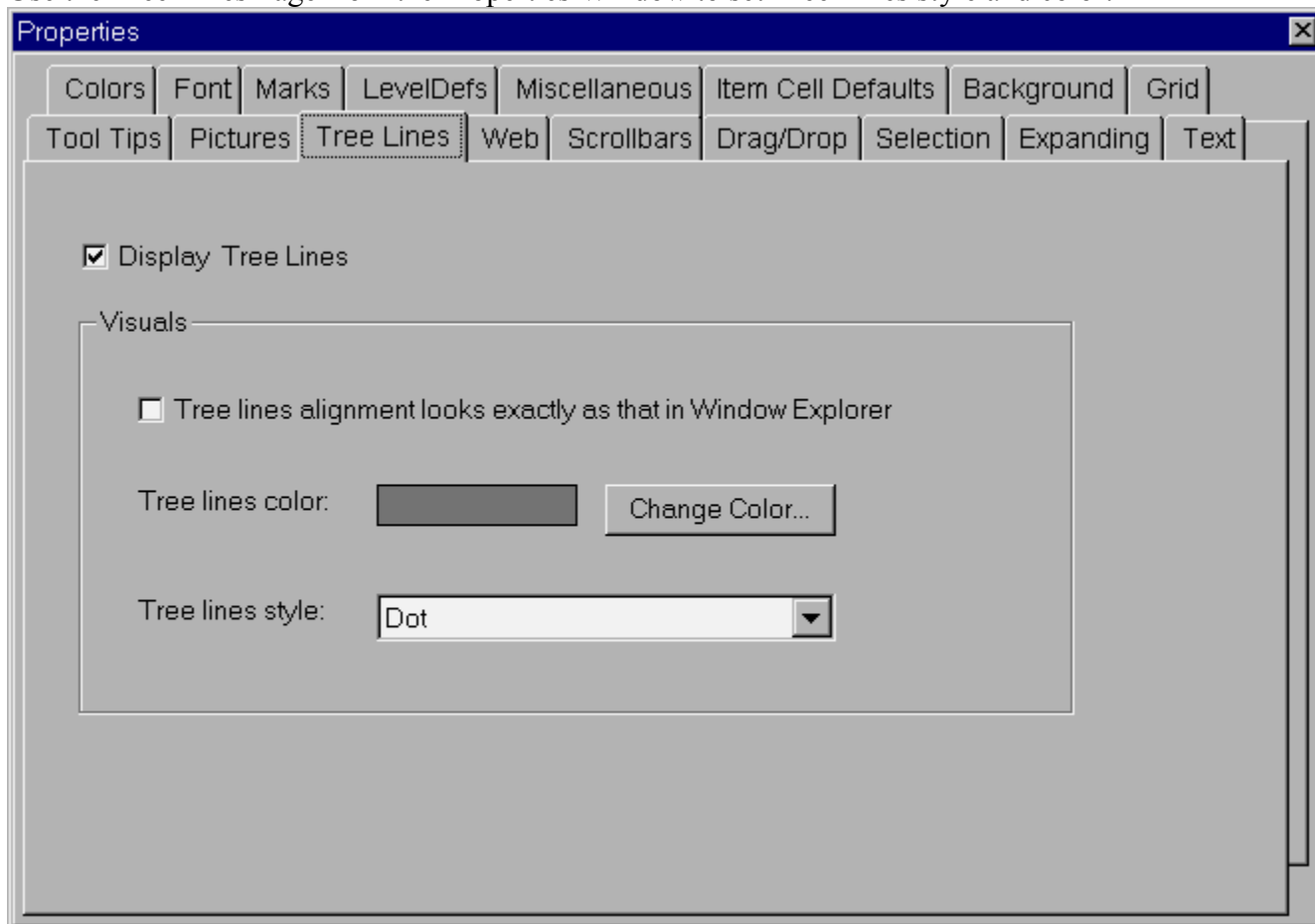
To add an additional picture to this list, use the Browse button.

[See also:](#)

TDesigner Features List

## Modifying Tree Line Settings

Use the Tree Lines Page from the Properties Window to set Tree Lines style and color:



Check the *Display Tree Lines* checkbox to turn on displaying of tree lines (use **ViewStyle** property to do this at run time).

Check the *Tree lines alignment looks exactly as that in Window Explorer* checkbox to display tree lines similar Window Explorer (use **ExplorerCompatible** property to do this at run time).

The *Tree lines color* panel displays tree lines color currently selected. In response to the *Change Color* button the Change Color dialog appears. Select the color you want and choose *OK* (use **TreeLinesColor** property to do this at run time).

Select the style you need in the *Tree Lines Style* combobox (use **TreeLinesStyle** property to do this at run time).

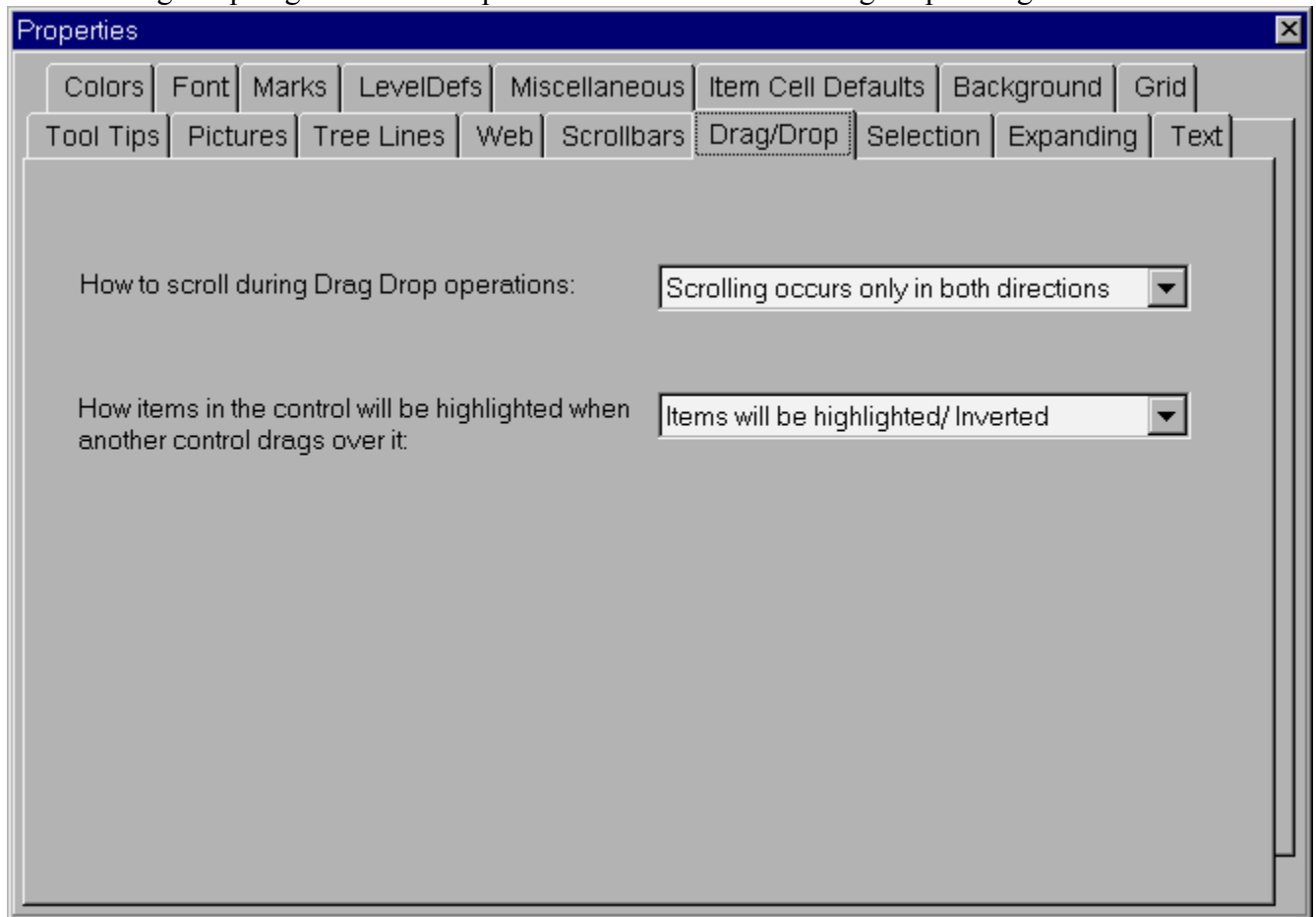
Use **ViewStyleEx**, **TreeLinesColor**, **TreeLinesStyle**, and **ExplorerCompatible** properties to control these settings at run time.

[See also:](#)

TDesigner Features List

## Specifying Drag Drop Settings

Use the Drag Drop Page from the Properties Window to control drag/drop settings:



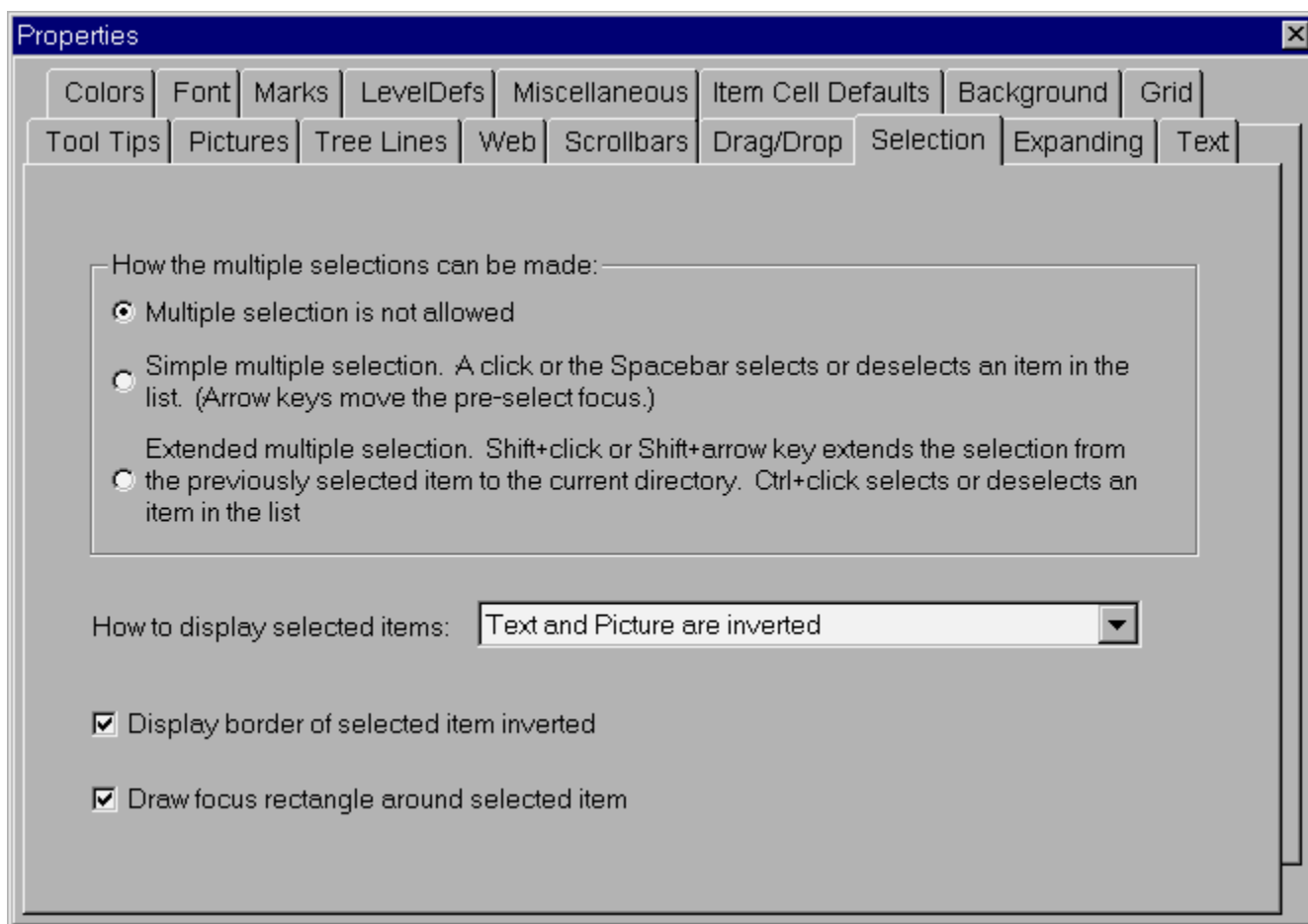
Select the drag drop settings you want in the *How to scroll during Drag Drop operations* and *How items in the control will be highlighted when another control drags over it* comboboxes (use **AutoScrDuringDragDrop** and **DragHighlight** properties to change these settings at run time).

[See also:](#)

TDesigner Features List

## Controlling Selection

Use the Selection Page from the Properties Window to change the way TList displays selected items:



Select the multiple selection type you want (use **MultiSelect** property to do this at run time).

Specify the desired style of for displaying selected items using the *How to display selected items* combobox (use **InvStyle** property to do this at run time).

The *Display border of selected item inverted* checkbox controls how the border border of selected items is displayed (use **InvBorderStyle** property to do this at run time).

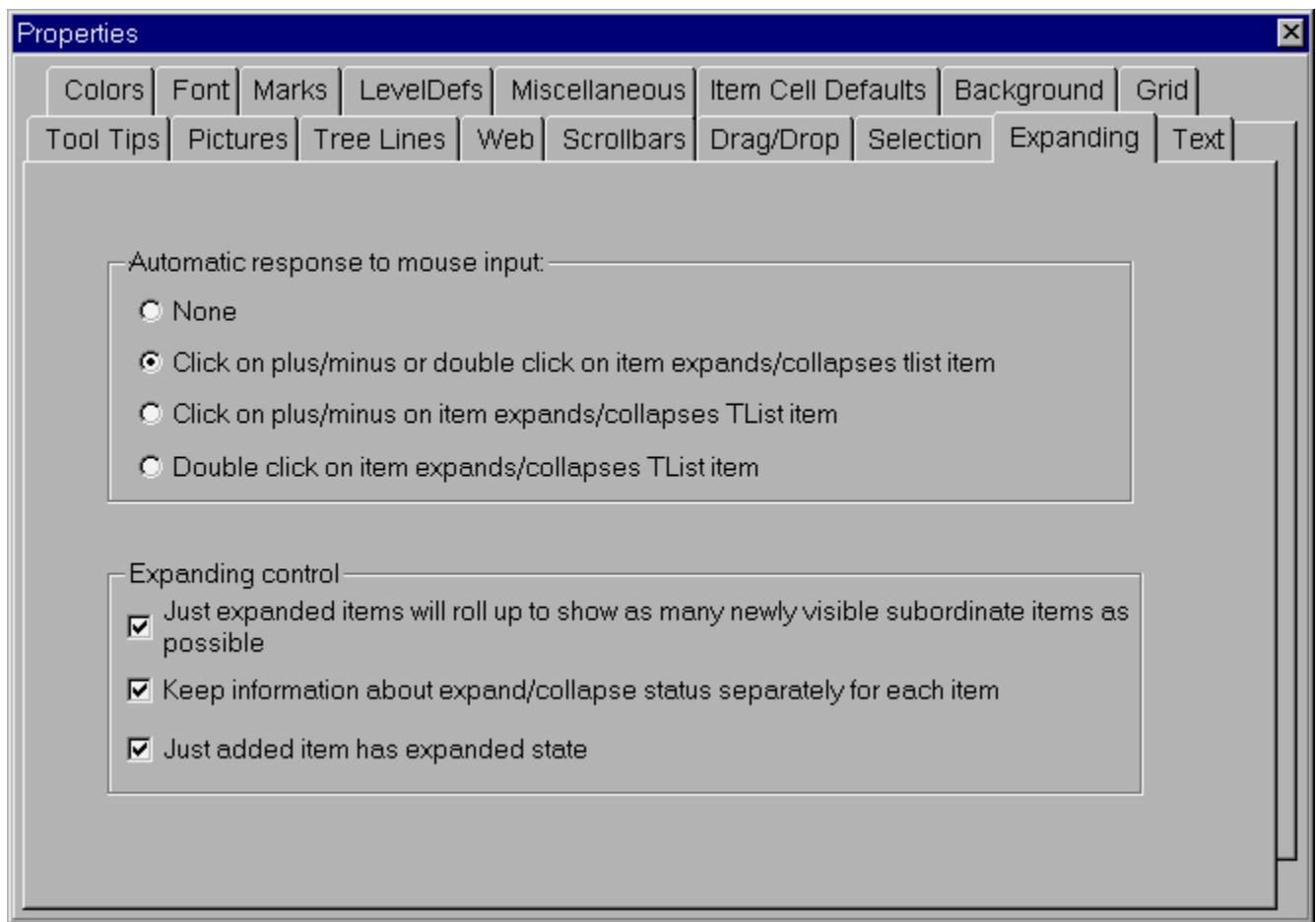
Check the *Draw focus rectangle around selected item* checkbox to display the focus rectangle around selected item (use **DrawFocusRect** property to do this at run time).

#### See also:

TDesigner Features List

## Controlling Expanding/Collapsing

Use the Expanding Page from the Properties Window to change the way TList expands and collapses items:



Select the *Automatic response to mouse input* type you want (use the **AutoExpand** property to do this at run time).

Check the expanding settings you need (use the **ShowChildren**, **ExpandChildren** and **ExpandNewItem** properties to do this at run time).

[See also:](#)

TDesigner Features List

## Controlling Text Display

Use the Text Page from the Properties Window to specify how TList displays item text:

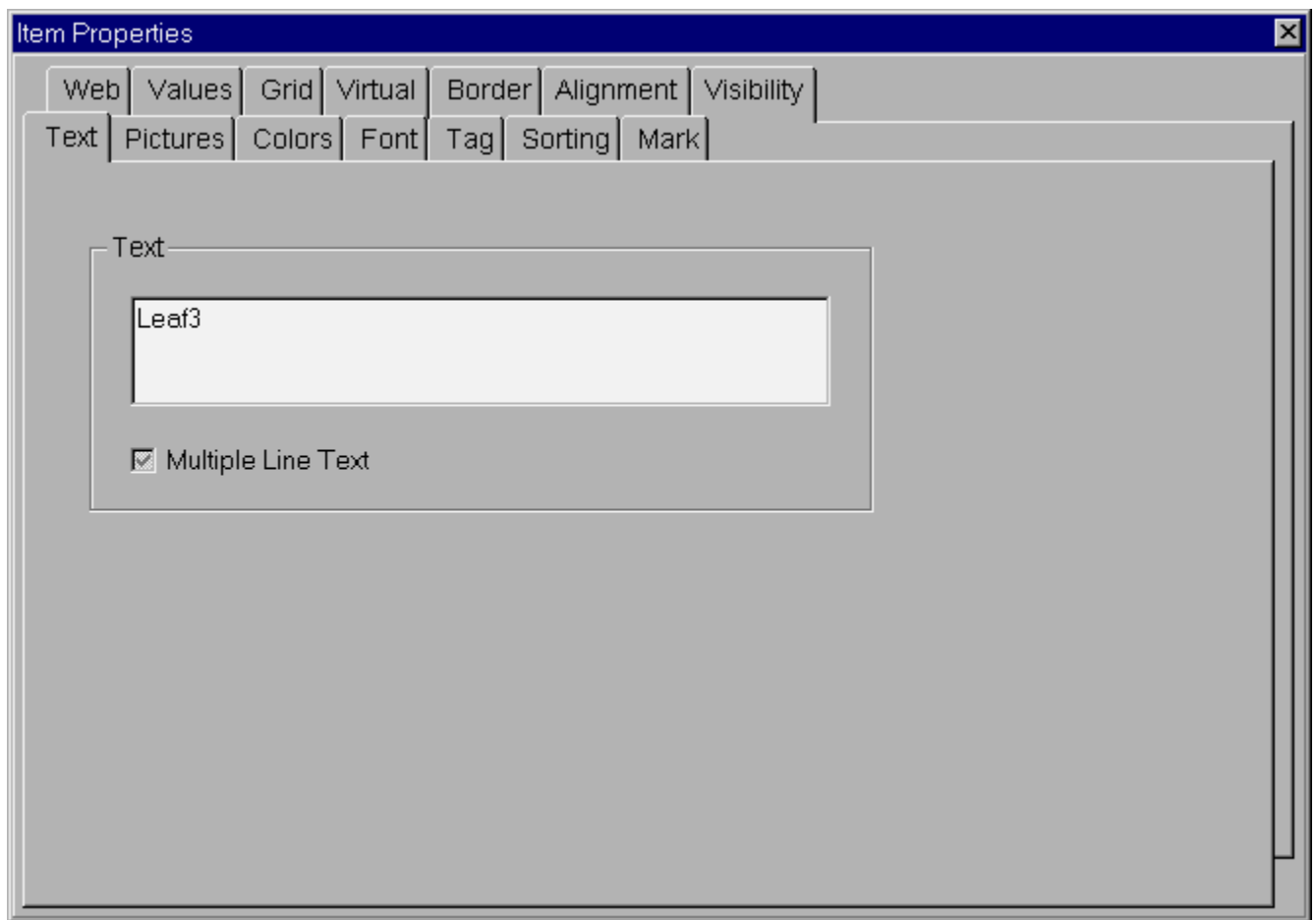
Check the *Display Text* checkbox to turn on displaying of item text (use **ViewStyle** property to do this at run time).

Specify the desired picture location for items which have multiline text using the *Where to display picture in the item which has multiple lines text* combobox (use the **PicInMultiLine** property to do this at run time).

Check the *Text of new added item is multiple lines text* checkbox to indicate that items will word wrap by default (use **DefMultiLine** property to do this at run time).

Specify width settings of mutiple line text items using the textboxes (use **WidthOfText** and **WidthOfTextMin** properties to do this at run time).

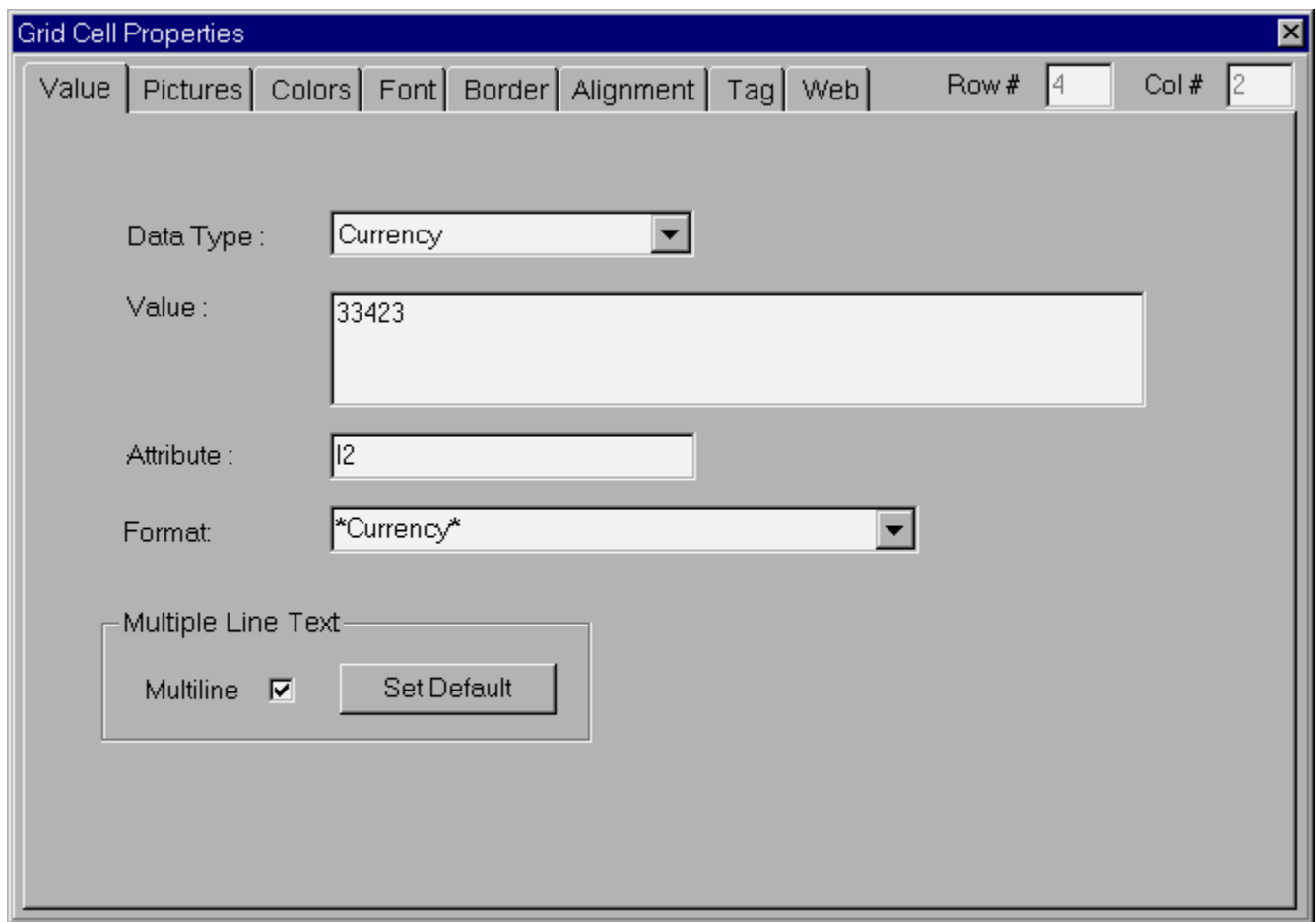
To set the text for each item, use the Text Page from the Item Properties Window:



Check the *Multiple Line Text* checkbox to specify the item text as multiline (use **ItemMultiLine** property to do this at run time).

Input text for the item into *Text* textbox (use the **List** property to change this setting at run time).

To set the text for each grid cell, use the Value Page from the Grid Cell Properties Window:



Select the desired value type for the grid cell using *Data Type* combobox.

---

Note You can use only String data type for the 1<sup>st</sup> column cells.

---

Specify attributes for the grid cell in *Attribute* textbox.

Select one of predefined format strings or specify your own format string for the grid cell using the *Format* combobox.

Check the *Multiple Line Text* checkbox to specify grid cell text as multiline.

Use the Value and **ValueName** properties of the GridCell's Value and **CellDef** objects, to change these settings at run time.

**See also:**

TDesigner Features List

## Controlling Fonts

Use Font Page from Properties Window to change font settings for the entire TList:

Use **FontBold**, **FontItalic**, **FontStrikethru**, **FontUnderline**, **FontName**, **FontSize** properties to change these settings at run time.

Use the Font Page from the Item Properties Window to change the font for the currently selected item in the Tree Window item:

Use **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, **ItemFontUnder**, **ItemFontName** and **ItemFontSize** properties to change these settings at run time.

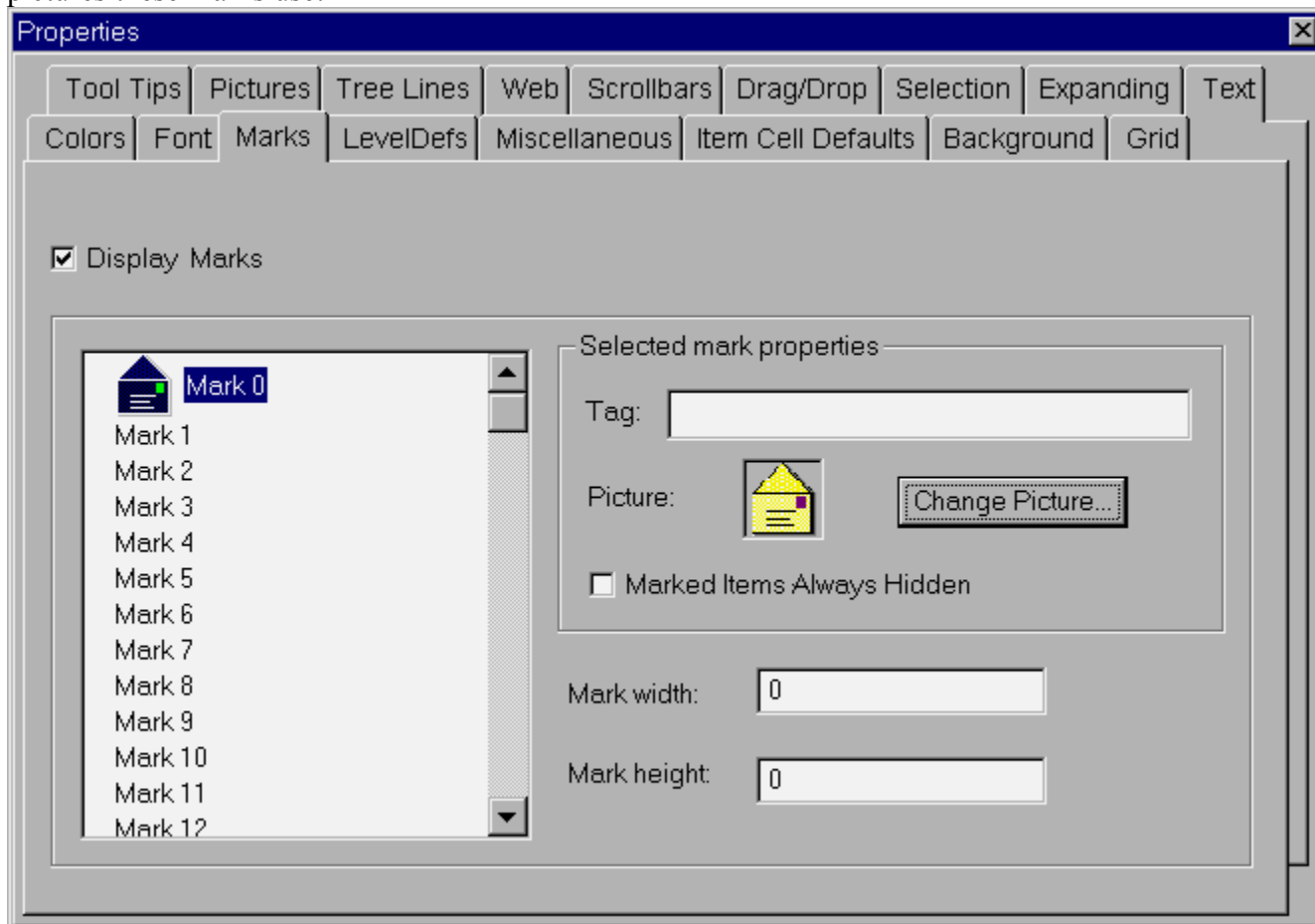
Use the Font Page from the Grid Cell Properties Window to change the font for the currently selected grid cell in the Tree Window item:

Use the **Font** property of GridCell's **CellDef** object to change this setting at run time.

In response to the *Change Font* button the standard Font dialog appears. Select the font you want and click *OK* button.

## Controlling Marks

Use the Marks Page from the Properties Window to specify how TList displays *marks* and what pictures these marks use:



Select the mark you wish to define in the mark list.

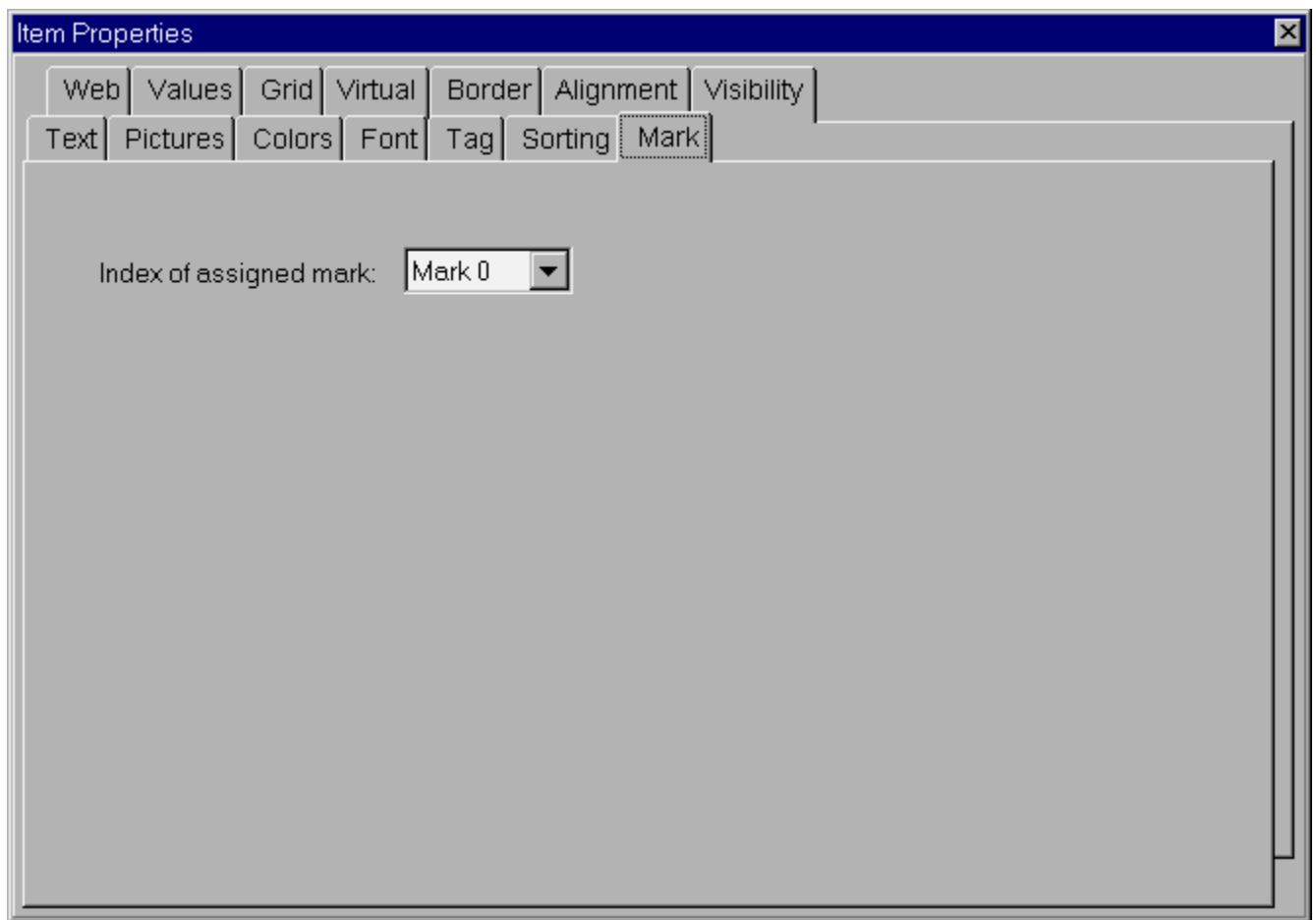
Specify a picture for this mark using *Change Picture* button (use **MarkPicture** property to do this at run time).

Specify the tag for this mark using *Tag* textbox (use **MarkTag** property to do this at run time).

Check *Marked Items Always Hidden* checkbox to indicate that all the items having this mark should be hidden (use **MarkedItemsAlwaysHidden** property to do this at run time).

Specify width and height of mark picture in *Mark width* and *Mark height* textboxes to stretch the picture to these sizes (use **MarkWidth**, **MarkHeight** properties to do this at run time).

Use the Mark Page from the Item Properties Window to associate an item with a mark:



Select the mark you need to specify for the item using *Index of assigned mark* combobox. Index 0 means no mark associated with the item.

Use **ItemMark** properties to change these settings at run time.

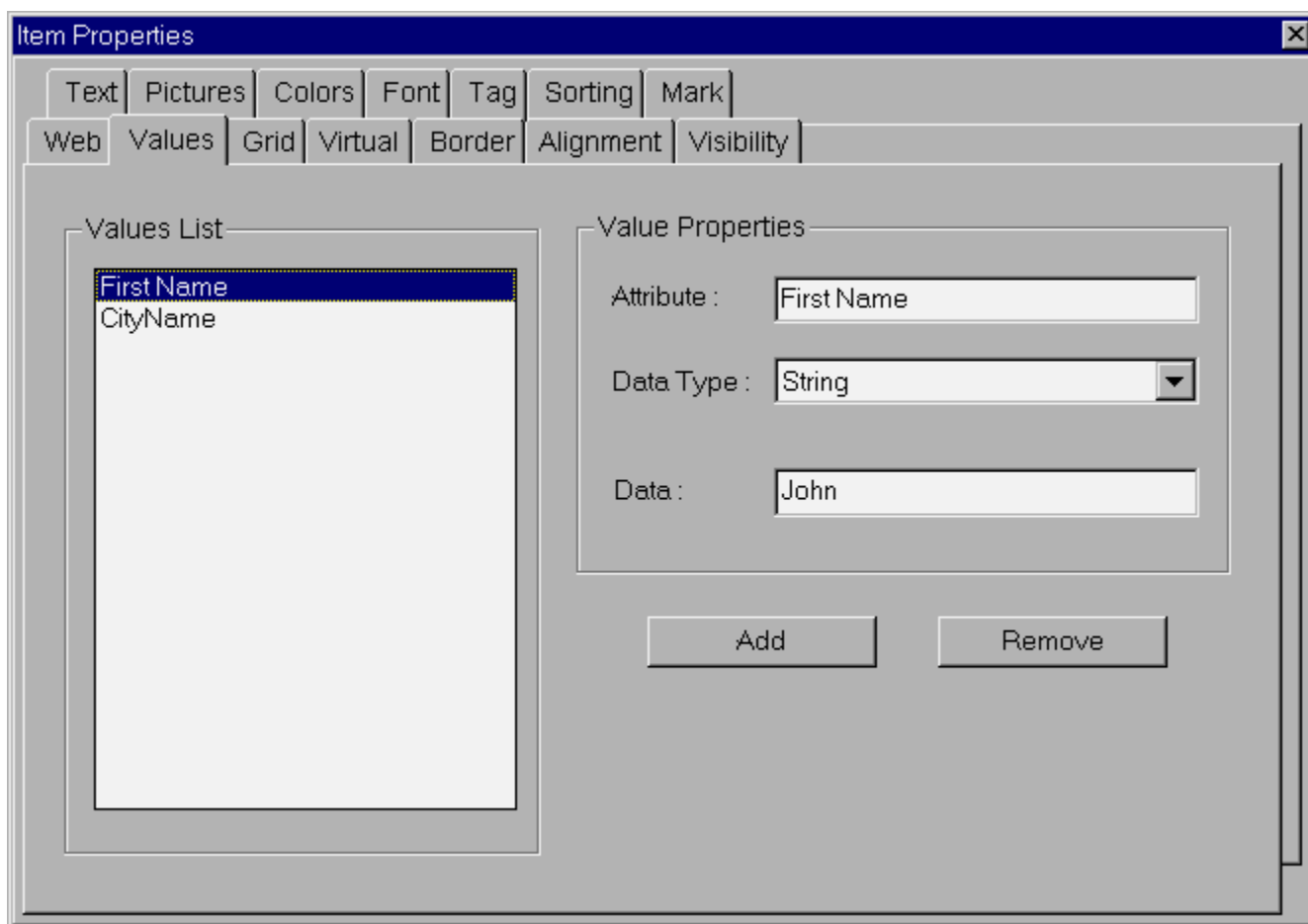
[See also:](#)

TDesigner Features List

## **Associating Additional Data with an Item**

Use the Values Page from the Item Properties Window to associate additional data with an item:





You can specify a new Value for the currently selected in the Tree Window item using *Add* button. Specify the attribute (value name) of the added Value in *Attribute* textbox. Select the data type you want to store in the Value using *Data Type* combobox. When you select Picture data type, you can select a picture using *Change Picture* dialog, otherwise you can specify a Value data in the *Data* textbox.

You can remove a Value from item's Values collection using *Remove* button.

Use **ItemValues** object collection for accessing a desired Value object at run time.

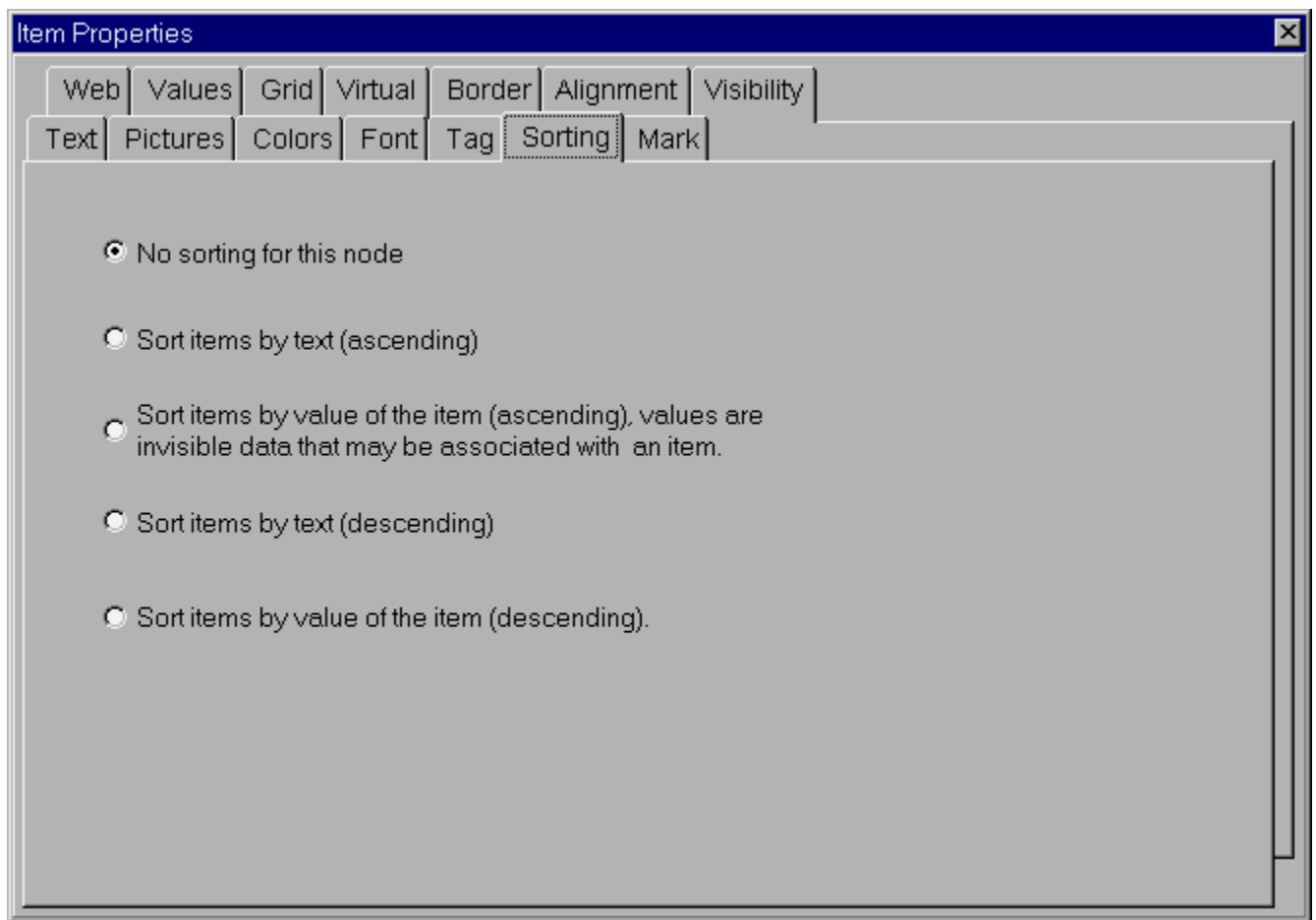
Use **ValueName**, **Value**, **ItemIndex** properties of a Value object to change these settings at run time.

**See also:**

TDesigner Features List

## Specifying Sorting Method

Use the Sorting Page from the Item Properties Window to specify the sorting method to be applied for children of an item:



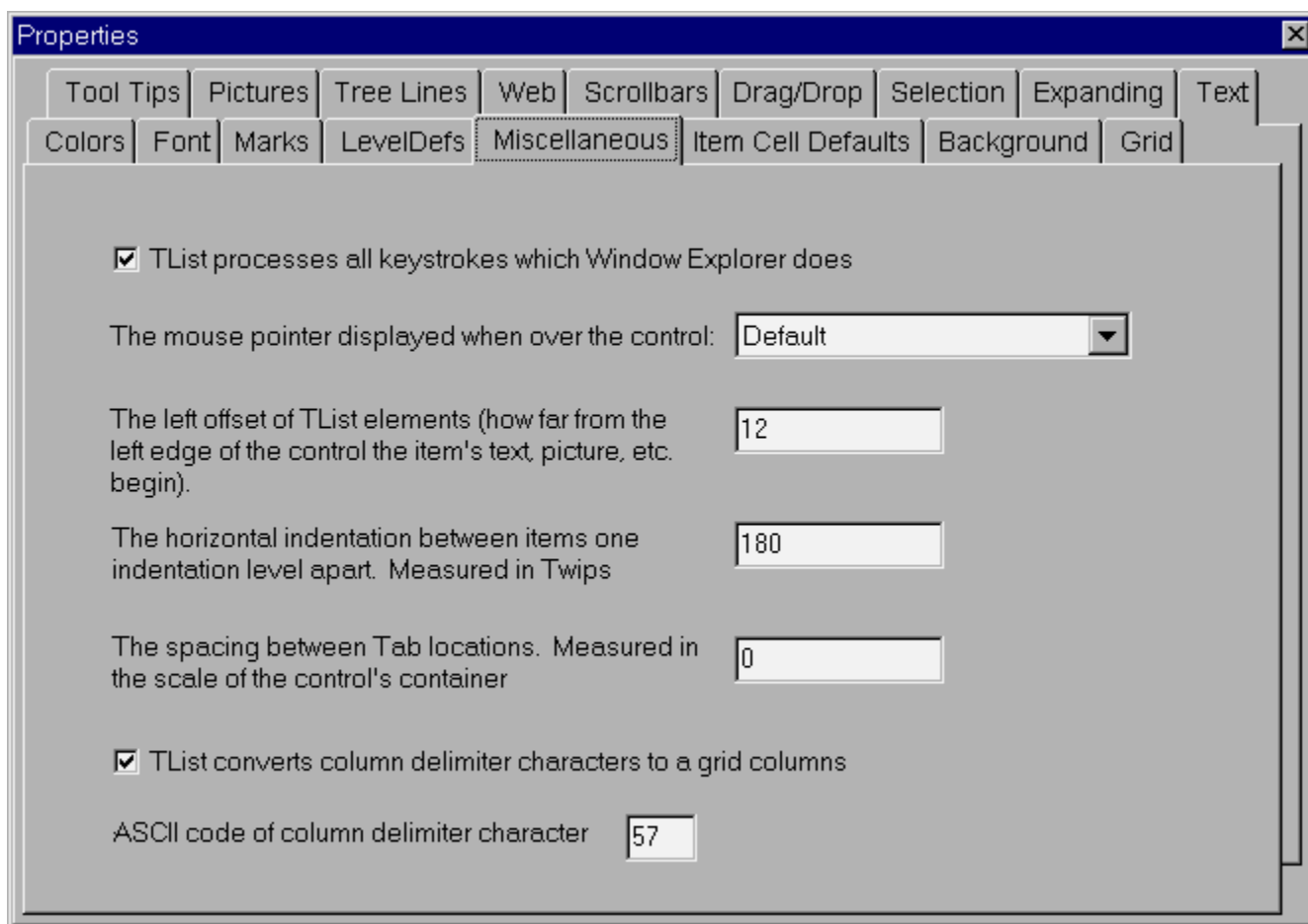
Use **ItemSorted** property to change these settings at run time.

[See also:](#)

TDesigner Features List

## Controlling Miscellaneous Settings

Use the Miscellaneous Page from the Properties Window to specify what mouse pointer is displayed , what keystrokes TList processes and other TList settings:



Check *TList processes all keystrokes which Window Explorer does* checkbox to force TList process keystrokes in the same way as Window Explorer.

Select the type of mouse cursor to display inside the TList using *The mouse pointer displayed when over the control* .

*The spacing between Tab locations*, *The left offset of TList elements* and *The horizontal indentation between items* textboxes allow further specification of TList's appearance.

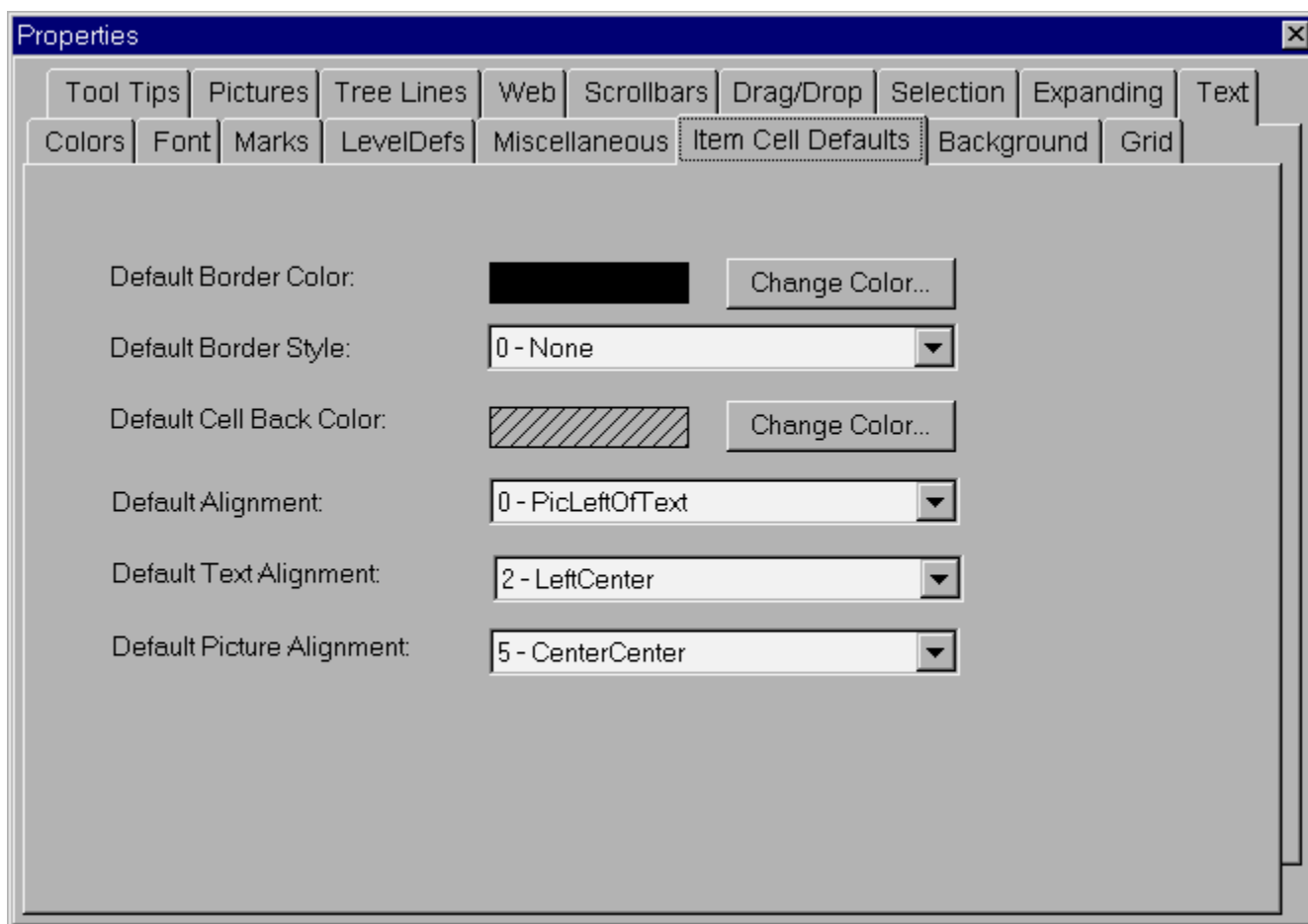
Use **ExplorerCompatible**, **MousePointer**, **ShiftStep**, **XOffset**, **TabStopDistance** properties to change these settings at run time.

#### See also:

TDesigner Features List

## Controlling Item Cell Default Settings

Use the Item Cell Defaults Page from the Properties Window to specify default item cell background and border color, border style and alignments.



Select default cell border color and cell background color from *Change Color* dialog using *Change Color* buttons (use **DefItemCellBorderColor** and **DefItemCellBackColor** properties to do this at run time).

Select cell border style using *Default Border Style* combobox (use **DefItemCellBorderStyle** property to do this at run time).

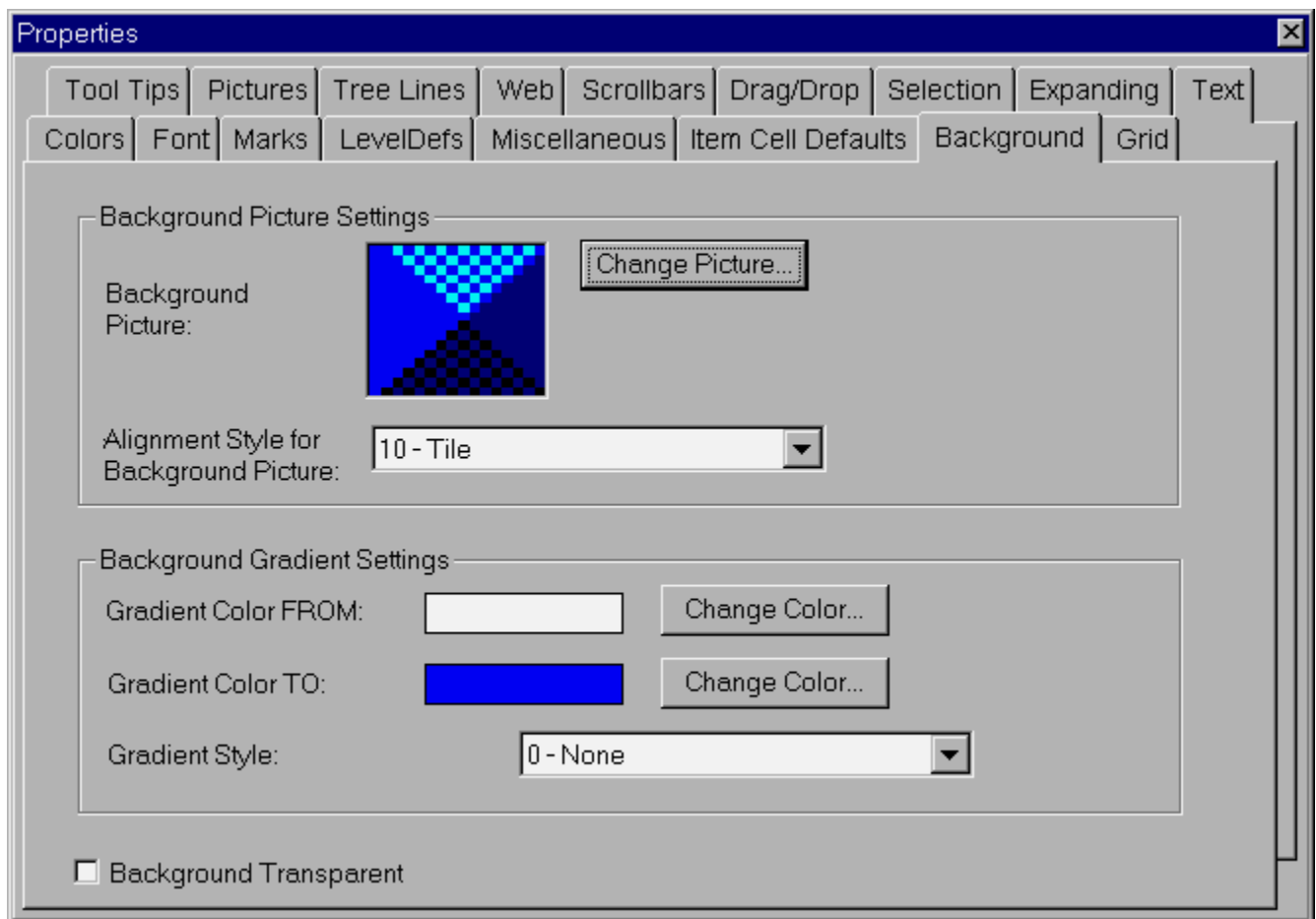
Select cell alignment using *Default Alignment*, *Default Text Alignment* and *Default Picture Alignment* comboboxes (use **DefItemCellAlignment**, **DefItemCellTextAlignment** and **DefItemCellPictureAlignment** properties to change these settings at run time).

#### See also:

TDesigner Features List

## Setting up Background

Use the Background Page from the Properties Window to specify background picture and background gradient settings.



Select background picture using *Change Picture* button (use **BackPicture** property to do this at run time).

Check *Background Transparent* checkbox to specify a transparent background (use **TransparentBackground** property to do this at run time).

Select background picture alignment using *Alignment Style for Background Picture* combobox (use **BackPictureAlignment** property to do this at run time)

Select style for gradient background using *Gradient Style* combobox (use **GradientStyle** property to do this at run time)

Select gradient background start and end colors from *Change Color* dialog using *Change Color* buttons (use **GradientColorFrom** and **GradientColorTo** properties to change these settings at run time).

[See also:](#)

TDesigner Features List

## Specifying Scrollbar Appearance

Use the Scrollbars Page from the Properties Window to specify whether to display scrollbars in TList.

Select which scrollbars (horizontal, vertical or both) should be displayed using *How to display scrollbars* combobox (use **ScrollBars** property to do this at run time)

Check *Always show scrollbars* checkbox to always show scrollbars (use **DisableNoScroll** property to do this at run time).

[See also:](#)

## Setting up Item And Grid Cell Borders

Use the Border Page from the Item Properties Window to specify the border for the currently selected item in the Tree Window item.

Select item border color from *Change Color* dialog using the *Change Color* button (use **ItemCell(ItemIndex&).BorderColor** property to do this at run time).

Select item border style using *Border Style* combobox (use **ItemCell(ItemIndex&).BorderStyle** property to do this at run time).

Use the Border Page from the Grid Cell Properties Window to specify the border for the currently selected item in the Tree Window grid cell.

Select the grid cell border color from *Change Color* dialog using the *Change Color* button (use **Cells(Row&,Col&).CellDef.BorderColor** property to do this at run time).

Select the grid cell border style using the *Border Style* combobox (use **Cells(Row&,Col&).CellDef.BorderStyle** property to do this at run time).

[See also:](#)

TDesigner Features List

## Setting up Item And Grid Cell Alignment

Use the Alignment Page from the Item Properties Window to specify alignment for the currently selected item in the Tree Window.

Select item cell alignment using *Alignment*, *Text Alignment* and *Picture Alignment* comboboxes (use **ItemCell(ItemIndex&).Alignment**, **ItemCell(ItemIndex&).TextAlignment** and **ItemCell(ItemIndex&).PictureAlignment** properties to change these settings at run time).

Use the Alignment Page from the Grid Cell Properties Window to specify alignment for the currently selected grid cell in the Tree Window.

Select grid cell alignment using *Alignment*, *Text Alignment* and *Picture Alignment* comboboxes (use **Cells(ItemIndex&).CellDef.Alignment**, **Cells(ItemIndex&).CellDef.TextAlignment** and **Cells(ItemIndex&).CellDef.PictureAlignment** properties to change these settings at run time).

[See also:](#)

TDesigner Features List

## Specifying Virtual Items

Use the Virtual Page from the Item Properties Window to specify whether the currently selected item in the Tree Window item has virtual children.

Check the *Virtual ON/OFF* checkbox to declare that the item has virtual children items and to specify the number of these virtual items in *Virtual Items Count* textbox (use **ItemVirtual** and **ItemVirtualCount** properties to change these settings at run time).

[See also:](#)

TDesigner Features List

## Setting up Item Visibility

Use the Visibility Page from the Item Properties Window to specify whether the currently selected item in the Tree Window item is always hidden when **ShowHiddenItems** button on main toolbar is unchecked (i.e. **ShowHiddenItems** property is set to False).

Check *Item Always Hidden* checkbox to make the currently selected in the Tree Window item always invisible.

Use **ItemAlwaysHidden** and **ShowHiddenItems** properties to change these settings at run time.

[See also:](#)

TDesigner Features List

## Setting up Item And Grid Cell Tag

Use the Tag Page from the Item Properties Window to specify a tag for the currently selected item in the Tree Window item.

Use the Tag Page from the Grid Cell Properties Window to specify a tag for the currently selected grid cell in the Tree Window.

Use **ItemTag** property of TList and **Tag** property of GridCell's **CellDef** object to change this setting at run time.

[See also:](#)

TDesigner Features List

## Setting up LevelDefs

Use the LevelDefs Page from the Properties Window to specify **LevelDef** default properties for each tree level.

Use **PictureOpen**, **PictureClosed**, **PictureLeaf** and **Indentation** properties of a desired **LevelDef** object to change these settings at run time.

[See also:](#)

TDesigner Features List

## Specifying a Tree Grid

Use the Grid Page from the Properties Window to specify the grid for the whole tree.

Use **TList.Grid** object's properties to change these settings at run time.

[See also:](#)

TDesigner Features List

## Specifying Item Grids

Use the Grid Page from the Item Properties Window to specify the grid for the currently selected item in the Tree Window.

Use **TList.ItemGrid(ItemIndex&)** object's properties to change these settings at run time.

[See also:](#)

TDesigner Features List

## Properties You Cannot Set with TDesigner

**ItemXXXValues** properties (**ItemIntValue**, etc) are obsolete; use **ItemTag** instead.

The **PictureType** property is not available since it is obsolete.

[See also:](#)

TDesigner Features List





# Objects reference

---

## Introduction

TList allows the programmer to work with the control in an object-oriented manner by exposing many of the interface elements as objects. The table below identifies the objects exposed by TList.

Object	Description
<b>TList</b>	The TList object references the control and provides control over the general look and feel of the control, as well as access to all other TList objects.
<b>TListCellDef</b>	The TListCellDef object holds settings for Items (list or tree rows, or grid cells), column captions, row headers etc. It also controls common graphic elements such as background color, foreground color etc.
<b>TListCheckBox</b>	The TListCheckBox object controls the editing / data entry behavior of grid or tree cells. The object provides support for both 2 and 3 state checkboxes.
<b>TListColDef</b>	The TListColDefs object represents a specific column of the TListGrid object
<b>TListColDefs</b>	The TListColDefs object holds a collection of TListColDef objects, representing all columns of the TListGrid object
<b>TListComboBox</b>	The TListComboBox object is used for convenient data editing by selecting a value from a list
<b>TListComboItem</b>	The TListComboItem object represents an item in a TListComboBox list (all items in the drop-down list).
<b>TListComboItems</b>	The TListComboItems object holds a collection of TListComboItem objects.
<b>TListDataObject</b>	The TListDataObject object is a container for data being transferred from an OleDragDrop source to a TList.
<b>TListDataObjectFiles</b>	The TListDataObjectFiles object is a collection whose elements represent a list of all filenames used by a TListDataObject object (such as the names of files that a user drags to or from the Windows File Explorer.).
<b>TListDateTime</b>	The TListDateTime object controls the editing /

	data entry behavior of grid or tree cells. The object provides support for setting the earliest and latest valid dates, date display format, and the use of a drop down calendar for date selection.
<b>TListEditInfo</b>	The TListEditInfo object provides convenient access to the in-place editing interface of the TList control. This object holds the editing style and provides an access to the particular type edit object.
<b>TListEditingChangeInfo</b>	The TListEditingChangeInfo object provides convenient access to the data the user enters using built-in editors (TListTextBox, TListComboBox, TListCheckBox, etc)
<b>TListGrid</b>	The TListGrid object provides convenient access to the data and all other interfaces of a specific grid object created inside TList control.
<b>TListGridCell</b>	The TListGridCell object defines data and formatting style for a cell of a grid.
<b>TListLevelDef</b>	The TListLevelDef object is used to specify formatting and other settings for a particular level of the tree.
<b>TListLevelDefs</b>	The TListLevelDef object is a standard collection that holds a series of TListLevelDefs objects.
<b>TListNode</b>	The TListNode Object provides simple and convenient access to all properties describing and controlling an item.
<b>TListNodes</b>	The TListNodes Object is a standard collection that holds a series of TListNode Objects.
<b>TListReport</b>	The TListReport object provides user with printing interface that allows to have full control over the printed report structure.
<b>TListPage</b>	The TListPage object provides all the information about the current page during printing process.
<b>TListPages</b>	The TList pages object is a collection of TListPage objects.
<b>TListRowDef</b>	This object represents a row in a <b>TListGrid</b> object. It provides accesses to get/set information specific for a particular row of the grid object.
<b>TListRowDefs</b>	The collection that holds a series of <b>TListRowDef</b> objects representing all rows in the grid.
<b>TListSpin</b>	The TListSpin object controls the editing / data entry behavior of grid or tree cells. The object displays spin buttons during editing allowing the user to increment or decrement the value by a discrete amount when clicking on the spin buttons.
<b>TListTextBox</b>	The TlistTextbox object is used for convenient data editing. Its properties and methods manage the the editing window layout/presentation.
<b>TListValue</b>	The TListValue object provides set/get

	interface to the hidden and visible data of the item or grid cell.
<b>TListValues</b>	The TListValues object is a standard collection object that holds a series of TListValue objects.
<b>TListSelectedGridCells</b>	The collection that holds a series of <b>TListGridCell</b> objects representing all selected cells in the active grid.
<b>TListSelectedGridRows</b>	The collection that holds a series of <b>TListRowDef</b> objects representing all fully selected rows (where all cells in the row are selected) in the active grid.
<b>TListSelectedGridColumnns</b>	The collection that holds a series of <b>TListColDef</b> objects representing all fully selected columns (where all cells in the column are selected) in the active grid.
TListTooltip	The object, the properties of which specify the content and presentation of the tooltip window to be displayed.
TListShowTooltipArgs	The object that defines the content and presentation of a tooltip. This object is presented as a parameter of the ShowTooltip event, giving the programmer full control over tooltip appearance and behavior.
TListTooltipStyle	The object that defines the colors and text alignment used for display of a tooltip window.
TListTooltipWindow	The object that defines the location and size of a tooltip window.

---

Note: none of the objects exposed by TList can be created using the Visual Basic CreateObject or New statements. The reference to an object must be obtained through the corresponding control/object property or method.

---

## TList Object

Here are properties, methods, functions and events of TList object:

- Properties
- Methods
- Events
- Functions

### *Properties (TList object)*

Property	Description
<b>AutoDragMode</b>	This property enables automatic drag/drop operations for a given control.
<b>About</b>	Shows the 'About' box with copyright information at design-time.
<b>ActiveGrid</b>	Returns a reference to a Grid object whose cell was clicked last.
<b>Add</b>	Adds item(s) from a <i>tree buffer</i> to the end of the subordinate item(s) list of the item.
<b>Align</b>	Determines where on a form and in what size the list box can appear.

<b>Appearance</b>	Returns or sets the paint style of TList on an MDIForm or Form object at run time.
<b>AutoExpand</b>	Specifies the default reaction of TList on mouse clicks and double clicks.
<b>AutoScrDuringDragDrop</b>	Determines whether to scroll TList during drag-drop operations.
<b>BackColor</b>	Specifies background color displayed in each item.
<b>BackPicture</b>	Specifies Background picture for the control.
<b>BackPictureAlignment</b>	Specifies alignment for the background picture.
<b>BorderStyle</b>	Specifies border style of TList control.
<b>BottomIndex</b>	Returns the last visible item in the list.
<b>Caption</b>	Specifies the string that will be shown in the caption.
<b>ClearItem</b>	Removes all subordinate items from an item.
<b>Clipboard</b>	Copies tree item(s) to the Windows clipboard or pastes them from the clipboard.
<b>ColDelimiter</b>	Specifies a delimiter character used in AddItem and AddRow methods.
<b>ConvertTabsToCols</b>	Determines whether AddItem method automatically creates columns.
<b>CopyItem</b>	Copies an item with its subordinate items to the temporary buffer.
<b>CopyItemSub</b>	Copies an item's subordinate items to the temporary buffer called <i>tree buffer</i> .
<b>CopyOne</b>	Copies an item without its subordinate items to the temporary buffer called <i>tree buffer</i> .
<b>CopySelected</b>	Copies selected items with their subordinate items to the temporary buffer called <i>tree buffer</i> .
<b>CurrentIndexMethod</b>	Specifies the way in which items in the list are enumerated.
<b>CurrentParent</b>	Specifies the Parent whose children are enumerated in the list when using CurrentIndexMethod = TLSys_Level (=2).
<b>CurrentItemBM</b>	Specifies the bookmark of the CurrentItem, i.e.: the Parent whose children are enumerated in the list when using CurrentIndexMethod = TLSys_Level. (=2)
<b>DefItemCellAlignment</b>	Specifies the default item cell alignment of the picture and the text.
<b>DefItemCellBackColor</b>	Specifies the default item cell background color.
<b>DefItemCellBorderCo</b>	Specifies the default item cell border

<b>lor</b>	color.
<b>DefItemCellBorderStyle</b>	Specifies the default item cell border style.
<b>DefItemCellDef</b>	Specifies the default settings for all cells in the tree.
<b>DefItemCellPictureAlignment</b>	Specifies the default item cell picture alignment.
<b>DefItemCellTextAlignment</b>	Specifies the default item cell text alignment.
<b>DefMultiLine</b>	Determines the default setting for the ItemMultiLine property
<b>DisableNoScroll</b>	Determines whether to show disabled vertical and horizontal scroll bars for the control when the list is not large enough to require scroll bars.
<b>DragHighlight</b>	Determines whether to highlight items as they are being dragged over.
<b>DragIcon</b>	Determines the icon to be displayed as a pointer in drag-and-drop operations.
<b>DragIconStyle</b>	Determines the appearance of icon to be displayed as drag-and-drop operation pointer depending on the TList item being dragged.
<b>DragMode</b>	Determines manual or automatic dragging mode for a drag-and-drop operations.
<b>DrawFocusRect</b>	Specifies whether to draw a focus rectangle.
<b>DropTarget</b>	Identifies the item that is being dragged over.
<b>EditingMode</b>	Provides control over automatic cell and item editing process.
<b>Enabled</b>	Determines whether the control can be acted upon.
<b>Environment</b>	Specifies the development environment in which TList is being used.
<b>Expand</b>	Specifies whether an item is expanded.
<b>ExpandChildren</b>	Specifies the way in which the TList keeps information about each item's expand/collapse status.
<b>ExpandEx</b>	Expands/collapses all items
<b>ExpandNewItem</b>	Specifies the default setting of expand/collapse status for each newly-added item.
<b>ExpandToLevel</b>	Expands and collapses all tree branches up to the specified level.
<b>ExplorerCompatible</b>	Defines Windows Explorer Outline compatibility.
<b>File</b>	Manages the tree picture table when saving to a file.
<b>FixedSize</b>	Determines whether all items have the

	same height.
<b>FocusRectStyle</b>	Determines how focus rectangle will be drawn around items in the tree.
<b>Font</b>	Returns a default font object
<b>FontBold</b>	Determines whether text should default to Bold.
<b>FontItalic</b>	Determines whether text should default to <i>italic</i> .
<b>FontName</b>	Determines the name of the default font.
<b>FontSize</b>	Determines the size of the default font.
<b>FontStrikeThru</b>	Determines whether text should default to <del>FontStrikeThru</del> .
<b>FontUnderline</b>	Determines whether the default text style is <u>Underlined</u> .
<b>ForeColor</b>	Determines the default foreground color for each item.
<b>FullPath</b>	Returns the path to an item.
<b>FullItemString</b>	Return a delimited string containing the data from each column of a row concatenated using the ColDelimiter character to separate column values.
<b>GetArrayProperty, SetArrayProperty, GetArrayPropertyID</b>	Allow you to set/get array properties of the TList control by name of the property. - for FoxPro developers important as workaround to FoxPro limitation on array properties.
<b>GradientColorFrom</b>	Specifies what color will be used to paint a gradient on the background.
<b>GradientColorTo</b>	Specifies what color will be used to paint a gradient on the background.
<b>GradientStyle</b>	Specifies the way a gradient will be drawn on the background.
<b>Grid</b>	Returns the Tree grid object.
<b>HasGrid</b>	Determines whether TList has Tree Grid.
<b>HasSubItems</b>	Indicates whether an item has subordinate items.
<b>Height</b>	Specifies the height of the TList control.
<b>HelpContextID</b>	Specifies the context number of the Help topic associated with the control.
<b>Hwnd</b>	Returns a window handle for the control.
<b>Image</b>	Determines the picture to be displayed with an item.
<b>ImageStretch</b>	Determines the stretch mode with which pictures are displayed.
<b>Index</b>	Identifies the control in a control array.
<b>InvBorderStyle</b>	Determines whether a cell changes its border when selected.
<b>Insert</b>	Inserts item(s) from the temporary buffer before the specified item.

<b>Indent</b>	Specifies the hierarchic indentation level of an item.
<b>InsertItem</b>	Inserts an item before another item at the same level
<b>InvImage</b>	Specifies the image to be displayed for selected items.
<b>InvStyle</b>	Specifies how selected items are displayed.
<b>IsClipboardAvailable</b>	Determines whether the Clipboard currently holds information recognized by TList.
<b>IsItemVisible</b>	Determines TList item visibility
<b>ItemAlwaysHidden</b>	Specifies whether an item is hidden regardless its parent visibility and expanded state.
<b>ItemBackColor</b>	The background color associated with an item.
<b>ItemBM</b>	Returns a Bookmark for an item.
<b>ItemCell</b>	Returns a reference to a TListCellDef object. An ItemCell is the portion of a item containing its text and optional additional picture.
<b>ItemCheckboxValue</b>	Returns or sets a value that determines the state of a checkbox if an item is checked for a List / Tree item or first column of a grid.
<b>ItemEditText</b>	Initiates or terminates edit mode for an item.
<b>ItemFontBold</b>	Determines whether a specific item's text is Bold.
<b>ItemFontItalic</b>	Determines whether a specific item's text is <i>Italic</i> .
<b>ItemFontName</b>	The font associated with a specified item.
<b>ItemFontSize</b>	The size of the font for the specified item.
<b>ItemFontStrike</b>	Determines whether a specific item's text is <del>FontStrikeThru</del> .
<b>ItemFontUnder</b>	Determines whether a specific item's text is <u>Underlined</u> .
<b>ItemForeColor</b>	The color of text associated with an item.
<b>ItemGrid</b>	Returns a grid object for the specified item. Read-only.
<b>ItemHasGrid</b>	Determines whether an item has a grid.
<b>ItemHeight</b>	Returns or sets the height of an item.
<b>ItemImageDefHeight</b>	Specifies the height of pictures displayed with an item.
<b>ItemImageDefWidth</b>	Specifies the width of pictures displayed with an item.

<b>Item...Value</b>	Specifies additional data stored with item.
<b>ItemLastSubItemIndex</b>	Returns the index of the last subordinate item for the specified item.
<b>ItemMark</b>	Specifies the index of a mark that is associated with an item.
<b>ItemMultiLine</b>	Specifies whether an item can display multiple lines of text.
<b>ItemNextSibling</b>	Returns the index of the next item at the same indentation level and with the same parent.
<b>ItemParent</b>	Returns the index of the parent of an item.
<b>ItemParentBM</b>	Returns bookmark of an item's parent.
<b>ItemPMPicType</b>	Specifies whether to display a plus/minus picture for an item which doesn't have any children.
<b>ItemPrevSibling</b>	Returns the index of the previous item at the same indentation level and with the same parent
<b>ItemSorted</b>	Initiates, halts or resets the sorting.
<b>ItemSortingMethod</b>	Specifies ascending or descending sort order.
<b>ItemSortingKey</b>	Specifies what data (visible text or hidden ItemValues) should be used as the key for the sorting.
<b>ItemSortingStyle</b>	Provides additional control such as numeric or case sensitive sorting
<b>ItemTag</b>	Specifies a string tag associated with the specified item.
ItemTooltip	Specifies the Tooltip object to be applied to a list or tree item
<b>ItemValues</b>	Holds array of associated data values for each item.
<b>ItemVirtualParent</b>	Specifies whether children of an item are virtual.
<b>ItemVirtualCount</b>	Specifies the number of virtual children for an item.
<b>ItemURL</b>	Specifies the URL for an item. This URL is used when WebAutoNavigate property is enabled.
<b>KeyboardActivation</b>	Controls how the user navigates (moves the focus specifying the Active Cell) through the Tree Grid structure using the keyboard.
<b>LevelDefs</b>	Returns TListLevelDef object associated with a specified tree level.
<b>Left</b>	Determines the horizontal placement of TList within its container.
<b>List</b>	Specifies text to be displayed with items.
<b>ListCount</b>	Returns the number of indexed items.



<b>ListCountEx</b>	Returns the number of item's children.
<b>ListIndex</b>	Specifies the item that currently has the focus.
<b>LoadAndAdd</b>	Loads item(s) from a file.
<b>LoadAndInsert</b>	Loads item(s) from a file.
<b>MarkHeight</b>	Specifies the height of a mark displayed next to an item.
<b>MarkPicture</b>	Specifies a picture for each mark.
<b>MarkedItemsAlways Hidden</b>	Specifies visibility for all items whose ItemMark property is identical to mark index.
<b>MarkTag</b>	Specifies a tag for each mark.
<b>MarkWidth</b>	Specifies the width of a mark displayed next to an item.
<b>Modifications</b>	Enables or disables recently added TList features to provide backwards compatibility with older editions.
<b>MousePointer</b>	Determines the mouse pointer displayed when over the control.
<b>MouseIcon</b>	Determines the mouse pointer that is displayed when over the control. Can be either icon or cursor.
<b>MSOutlineAdd</b>	Determines the way that the AddItem method works.
<b>MultiSelect</b>	Specifies whether a user can make multiple selections.
<b>Name</b>	Specifies the name that must be used in code to refer to the list box.
<b>NewIndex</b>	Returns the index of the item which was used in the last operation.
<b>NoPictureRoot</b>	Determines how pictures next to root level items are displayed.
<b>OLEDragMode</b>	Specifies whether TList itself or the programmer manually handles an OLE drag/drop operation.
<b>OLEDropMode</b>	Specifies how a target TList component handles drop operations.
<b>Parent</b>	Returns the form in which the control is located.
<b>PathSeparator</b>	Sets and returns the item delimiter string used when accessing the FullPath property.
<b>PicInMultiLine</b>	Specifies positioning of a picture when displayed next to word wrapped items.
<b>PictureClosed</b>	Specifies the default closed picture for an item.
<b>PictureInverted</b>	Specifies the default inverted picture for an item.
<b>PictureLeaf</b>	Specifies the default leaf picture for an item.

<b>PictureMark</b>	Specifies the default mark picture for an item.
<b>PictureMinus</b>	Specifies the default minus picture for an item.
<b>PictureOpen</b>	Specifies the default open picture for an item.
<b>PicturePalette</b>	Determines the palette to be used to display all pictures.
<b>PicturePlus</b>	Specifies the default plus picture for an item.
<b>PictureRoot</b>	Specifies the default root picture for an item.
<b>PictureType</b>	Determines how default pictures are used.
<b>Redraw</b>	Controls repainting of the control.
<b>Report</b>	Returns a reference to the TListReport object, which controls the printing process from TList.
<b>Root</b>	Returns a TListNodes root collection containing all items at zero level.
<b>Save</b>	Saves item(s) to a file.
<b>SaveOne</b>	Saves item(s) to a file.
<b>SaveSub</b>	Saves item(s) to a file.
<b>Scrollbars</b>	Determines how scrollbars are displayed.
<b>ScrollHorz</b>	Scrolls the contents of TList horizontally.
<b>ScrollHPosition</b>	Specifies TList's horizontal scroll position in pixels.
<b>ScrollHRange</b>	Returns the maximum range of horizontal scrolling in pixels.
<b>ScrollVPosition</b>	Specifies the Vertical Scroll position of TList measured in visible items.
<b>ScrollVRange</b>	Returns the maximum range of Vertical scrolling.
ScrollVModeKeyboard	Controls TList's vertical scrolling behavior in response to keyboard Up/Down arrow keys
ScrollVModeMouse	Controls TList's vertical scrolling behavior in response to mouse clicks upon the Up/Down buttons within the vertical scrollbar.
ScrollVStepKeyboard	Controls the distance by which TList scrolls vertically when smooth scrolling in response to keyboard Up/Down arrow keys
ScrollVStepMouse	Controls the distance by which TList scrolls vertically when smooth scrolling in response to clicks on Up / Down buttons within TList scrollbar.
<b>SelBackColor</b>	Specifies the background color to be displayed for selected items.
<b>Selected</b>	Determines whether an item is selected.

<b>SelectEx</b>	Selects a group of items.
<b>SelForeColor</b>	Specifies the foreground color to be displayed for selected items.
<b>SelItemCount</b>	Returns the number of selected items.
<b>SelItemIndex</b>	Returns the indexes of selected items.
<b>Shift</b>	Specifies an item's hierarchic indentation.
<b>ShiftStep</b>	Specifies an item's horizontal indentation in terms of the container's scale.
<b>ShowCaption</b>	Determines the visibility of the caption.
<b>ShowChildren</b>	Determines whether expanded items will roll up to show as many subordinate items as possible.
<b>ShowHiddenItems</b>	Determines whether "always hidden items" are shown regardless of ItemAlwaysHidden and MarkedItemsAlwaysHidden properties settings.
<b>SmartDragDrop</b>	Determines when TList updates the Selected array.
<b>TabIndex</b>	Specifies the position within the tab sequence of controls on a form.
<b>TabStop</b>	Determines whether the control's focus can be reached by tabbing from other controls.
<b>TabStopDistance</b>	Specifies the spacing between Tab locations
<b>Tag</b>	Specifies a string associated with a TList control.
<b>Text</b>	Specifies the text of the selected item.
<b>ToolTipsBackColor</b>	Specifies the background color of the Tool Tip box.
<b>ToolTipsDelay</b>	Specifies the delay that will take place before showing ToolTips window.
<b>ToolTipsForeColor</b>	Specifies the text color of the Tool Tip box.
<b>ToolTipsMode</b>	Specifies whether ToolTips are shown while user is moving the mouse over an item.
<b>ToolTipsText</b>	Specifies a string to be shown as a tooltip whenever the mouse hangs for some time over the TList control itself.
<b>ToolTipsViewStyle</b>	Specifies which set of colors is used to paint ToolTips.
<b>Top</b>	The distance between the top edge of the list box and the top edge of its container.
<b>TopIndex</b>	Sets and returns the item that appears in the topmost position in the list box.
<b>TransparentBackground</b>	Determines whether to show TList with a transparent Background.

<b>TransparentBitmap</b>	Specifies whether bitmaps are displayed transparent.
<b>TransparentBitmapColor</b>	Determines a transparent color for bitmaps.
<b>TreeLinesColor</b>	Determines the color of tree lines.
<b>TreeLinesStyle</b>	Determines the style of tree lines.
<b>TreeLinesHighlightColor</b>	Specify highlight color used for 3-D tree lines.
<b>TreeLinesShadowColor</b>	Specify shadow color used for 3-D tree lines.
<b>TriggerEvents</b>	Controls what events to generate.
<b>Version</b>	Returns the current version of the control.
<b>ViewStyle</b>	Determines the way an item will be displayed.
<b>ViewStyleEx</b>	Determines the way an item will be displayed.
<b>Visible</b>	Determines whether the control is visible or hidden.
<b>VisualRoot</b>	Determines which portion of the tree to display.
<b>WebAutoNavigate</b>	Determines whether TList navigates through the Web automatically.
<b>WebTargetFrame</b>	Specifies a name of the frame in which to display the loaded Web document.
<b>WebURLBase</b>	Specifies the base for URL addresses stored in TList's items.
<b>Width</b>	The width of the control.
<b>WidthOfText</b>	Specifies the width of text for items which can display multiple lines of text.
<b>WidthOfTextMin</b>	Specifies the minimum width of multi line text.
<b>WheelScrolling</b>	Provides the developer with control over IntelliMouse functionality.
<b>XOffset</b>	Sets the left offset of TList items.

### ***Events (TList object)***

<b>Event</b>	<b>Description</b>
<b>AutoDragRequest</b>	Triggered at the start of drag/drop operation in AutoDragDrop mode.
<b>AutoDragComplete</b>	Triggered upon completion of any drag/drop operation in AutoDragDrop mode.
<b>AfterEditing</b>	Occurs after item text editing.
<b>BeforeItemDeactivate</b>	Occurs before the currently active item is deactivated by the user's mouse or keyboard action.
<b>BeforeGridRowDeactivate</b>	Occurs before the currently active grid row is deactivated by the user's mouse or keyboard action.

<b>BeforeGridCellDeactivate</b>	Occurs before the currently active grid cell is deactivated by the user's mouse or keyboard action.
<b>BeginPage</b>	Generated for each page as it is ready to be printed.
<b>Click</b>	Occurs when the user selects an item in a list box.
<b>Collapse</b>	Occurs when an item is collapsed.
<b>DbClick</b>	Occurs when the user double-clicks.
<b>DragDrop</b>	Occurs when a drag-and-drop operation is complete.
<b>DragOver</b>	Occurs when a drag-and-drop operation is in progress.
<b>DragDropEx</b>	Occurs immediately before OleDragDrop event when using TList's AutoDragDrop support.
<b>DragOverEx</b>	Occurs immediately before OleDragOver event when using TList's AutoDragDrop support.
<b>EditingKeyDown</b>	Occurs when the user presses a key while in text editing mode.
<b>EditingKeyPress</b>	Occurs when the user presses a key while in text editing mode.
<b>EditingKeyUp</b>	Occurs when the user releases a key while in text editing mode.
<b>EndPage</b>	Generated for each page after TList's data has been printed
<b>Expand</b>	Occurs when an item is expanded.
<b>GotFocus</b>	Occurs when the control receives the focus.
<b>GridCellActivate</b>	Occurs right after a cell becomes active (having focus), but before any changes are displayed on the screen.
<b>GridCellDeactivate</b>	Occurs when a cell is being deactivated (losing focus), but before the corresponding changes become visible on the screen.
<b>GridCellClick</b>	Occurs when the user selects a cell in a grid by clicking the mouse button.
<b>GridCellDbClick</b>	Occurs when the user double-clicks a cell in a grid.
<b>GridCellRequestEditing</b>	Occurs when editing is initiated in a grid cell
<b>GridCellAfterEditing</b>	Occurs when the <i>TListGridObject.CellEdit</i> property is set to TLEDITMODE_END or when editing is canceled or completed by the user.
<b>GridCellEditingKeyDown</b>	Occurs as a result of Keyboard actions during editing.
<b>GridCellEditingKeyUp</b>	Occurs as a result of Keyboard actions during editing.
<b>GridCellEditingKeyPress</b>	Occurs as a result of Keyboard actions during editing.

<b>GridCellEditingChange</b>	Occurs as a result of end-user editing changes using a built-in editors.
<b>GridRowActivate</b>	Occurs right after a row becomes active (having focus), but before any changes are displayed on the screen.
<b>GridRowDeactivate</b>	Occurs when a row is being deactivated (losing focus), but before the corresponding changes become visible on the screen.
<b>GridRowTitleClick</b>	Occurs when the user clicks a cell in a title row in any Grid object in TList.
<b>GridColumnTitleClick</b>	Occurs when the user clicks a cell in a column in any Grid object in TList.
<b>GridCornerTitleClick</b>	Occurs when the user clicks a corner cell in any Grid object in TList.
<b>ItemActivate</b>	Occurs after an item becomes active (gets the focus).
<b>ItemDeactivate</b>	Occurs right before an item becomes inactive (loses the focus).
<b>ItemClick</b>	Occurs when the user selects an item in a list box by clicking the mouse button.
<b>ItemDbClick</b>	Occurs when the user double-clicks a cell item in a grid.
<b>ItemQueryData</b>	Occurs when the TList needs a virtual item data.
<b>ItemEditingChange</b>	Occurs as a result of end-user editing changes using built-in editors.
<b>HScroll</b>	Occurs when the user scrolls the control horizontally.
<b>KeyDown</b>	Occurs when the user presses a key while the control has the focus.
<b>KeyPress</b>	Occurs when the user presses a key, after the KeyDown event.
<b>KeyUp</b>	Occurs when the user releases a key while an object has the focus.
<b>LostFocus</b>	Occurs when the control loses focus.
<b>MarkClick</b>	Occurs when the mark picture associated with an item is clicked.
<b>MarkDbClick</b>	Occurs when the mark picture associated with an item is double-clicked.
<b>MouseDown</b>	Occurs when the user presses a mouse button over the control.
<b>MouseMove</b>	Occurs when the user moves the mouse over the control.
<b>MouseUp</b>	Occurs when the user releases a mouse button over the control.
<b>MouseWheel</b>	Occurs when user rotates mouse wheel.
<b>OLEDragDrop</b>	Occurs when an OLE object is dropped into the control.
<b>OLEDragOver</b>	Occurs when an OLE object is dragged over the control.

<b>OLECompleteDrag</b>	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.
<b>OLEGiveFeedback</b>	Occurs after every <b>OLEDragOver</b> event and allows the source TList component to provide visual feedback to the user, such as changing the mouse cursor
<b>OLESetData</b>	Occurs on a source component when a target component performs the <b>GetData</b> method on the source's <b>TListDataObject</b> object, but before the data for the specified format has been loaded.
<b>OLEStartDrag</b>	Occurs when a component's <b>OLEDrag</b> method is called (explicitly or not).
<b>PictureClick</b>	Occurs when the picture associated with an item is clicked.
<b>PictureDbClick</b>	Occurs when the picture associated with an item is double-clicked.
<b>PlusMinusClick</b>	Occurs when the plus/minus picture associated with an item is clicked.
<b>PlusMinusDbClick</b>	Occurs when the plus/minus picture associated with an item is double-clicked.
<b>PreparePage</b>	Generated for each page after the PrepareForPrinting method has been called
<b>TitlesResize</b>	Generated when end-user starts to change a column width or row height by dragging a grid line.
<b>ShowToolTip</b>	Generated when a Tooltip is about to be displayed
<b>VScroll</b>	Occurs when the user scrolls the control vertically.
<b>RequestEditing</b>	Occurs when editing is initiated in a list or tree item ( not part of a grid, or in first column of grid)

### **Methods (TList object)**

<b>Method</b>	<b>Description</b>
<b>AddAfter</b>	Inserts a new item immediately after an item specified by its index.
<b>AddItem</b>	Adds an item to the list.
<b>AddItem2</b>	Simultaneously adds item and sets item formatting.
<b>AddItem2Ex</b>	Simultaneously adds item and sets item formatting.
<b>BeforeDrag</b>	Prepares TList for Drag Drop.
<b>Clear</b>	Removes all items from a TList.
<b>CopyBuffer</b>	Copies a <i>tree buffer</i> to the Windows clipboard.

<b>Drag</b>	Begins, ends, or cancels dragging controls.
<b>FindItem</b>	Searches for an item(s) by its item text.
<b>FindValue</b>	Searches for an item(s) by its associated data (Item...Value).
<b>FreeBuffer</b>	Releases a temporary buffer.
<b>GetItemByXY</b>	Returns the index of an item at a given X/Y coordinate.
<b>GetItemRect</b>	Returns coordinates of the item specified by an index.
<b>HitTest</b>	Determines what object is under the cursor.
<b>IndexByBM</b>	Converts the bookmark of an item into a valid index.
<b>IsValidBM</b>	Checks the validity of a bookmark.
<b>IsValidBuffer</b>	Checks validity of a temporary buffer.
<b>LoadBuffer</b>	Loads a <i>tree buffer</i> from a file.
<b>LoadData</b>	Loads items and properties settings from a .TLT data file. This file can be prepared by TDesigner application or programmatically via SaveData TList method.
<b>LoadPicture</b>	Loads picture from file and returns pointer on it.
<b>Move</b>	Moves a TList control.
<b>MoveItem</b>	Moves an item to a specified position in the tree.
<b>Node</b>	Returns a TListNode ObjectObject that corresponds to an item specified by its index.
<b>OLEDrag</b>	Initiates an OLE drag/drop operation.
<b>OnDragDrop</b>	Prepares TList to accept DragDrop events.
<b>OnDragOver</b>	Prepares TList to accept DragOver events.
<b>PasteBuffer</b>	Copies information from the Windows clipboard into a <i>tree buffer</i> .
<b>PrintOneStep</b>	Prints TList contents
<b>Refresh</b>	Forces an immediate repaint or update of the control.
<b>RefreshItems</b>	Forces TList to generate ItemQueryData for the specified virtual items.
<b>RemoveItem</b>	Removes an item.
<b>SaveBuffer</b>	Saves a <i>tree buffer</i> to a file.
<b>SaveData</b>	Saves TList items and property settings in specified .TLT data file.
<b>SetFocus</b>	Sets the focus to a TList control.
<b>TranslateIndex</b>	Translates an index value from one indexing method to another.
<b>UpdateBackground</b>	Instructs TList to repaint its background (use when elements behind a Transparent TList are updated).
<b>WebGoBack</b>	Navigates to the previous item in the history list.
<b>WebGoForward</b>	Navigates to the following item in the



history list.

**WebNavigate**

Navigates to any document in Web by URL.

**ZOrder**

Places the control at the front or back of the z-order within its graphical level.

### Functions (TList object)

*\* Note that in the OCX editions, certain functions are implemented in the associated BAS modules provided for Visual Basic users and are not part of the control itself. In this case the OCX has a corresponding method which is called by the BAS module function. The BAS module function is provided solely to aid in conversion from VBX syntax.*

Function	Description
<b>TListCopyBuffer</b>	Copies a <i>tree buffer</i> to the Windows clipboard.
<b>TListFindItem</b>	Searches for an item(s) by its item text.
<b>TListFindValue</b>	Searches for an item(s) by its associated data.
<b>TListFreeBuffer</b>	Frees the memory associated with a <i>tree buffer</i> .
<b>TListGetItemByXY</b>	Returns the index of the item at a given X/Y coordinate.
<b>TListGetItemRect</b>	Returns the coordinates of the item specified by an index.
<b>TListIndexByBM</b>	Converts the bookmark of an item into a valid index.
<b>TListIsClipboardFormatAvailable</b>	Checks whether there is valid information for TList in the clipboard.
<b>TListIsValidBM</b>	Checks the validity of a bookmark.
<b>TListIsValidBuffer</b>	Checks the validity of a <i>tree buffer</i> .
<b>TListLoadBuffer</b>	Loads a <i>tree buffer</i> from a file.
<b>TListPasteBuffer</b>	Copies information from the Windows clipboard into a <i>tree buffer</i> .
<b>TListSaveBuffer</b>	Saves a <i>tree buffer</i> to a file.
<b>TListTranslateIndex</b>	Translates an index value from one indexing method to another.

---

## TListCellDef Object

### Description

A TListCellDef object holds settings for Items, column captions, row headers etc. It also controls common graphic elements such as background color, foreground color etc.

TList provides access to TListCellDef objects for individual items in a list or tree, individual grid cells, and to set defaults for all data, for a specific row or column, for a heirarchic level, for all cells in a grid.

### Properties

Property	Description
<b>Activatable</b>	Provides control over the end-user's ability to navigate to any particular item/cell/row using keyboard, mouse or using properties
<b>Alignment</b>	Specifies how picture and text are aligned in a cell.

<b>BackColor</b>	Specifies the background color for a cell.
<b>BorderColor</b>	Specifies the border color for a cell.
<b>BorderStyle</b>	Specifies the border type for a cell.
<b>EditInfo</b>	Specifies an object holding the editing style and providing access to the edit objects.
<b>Font</b>	Specifies text font for a cell. Font objects have Name, Underline, Italic... properties
<b>Font3D</b>	Controls the presentation of either standard or 3-D text appearance in TList.
<b>FontShadowColor</b>	Determine the shadow colors used to present 3D text for normal items/cells/rows/columns.
<b>FontShadowSelectedColor</b>	Determine the shadow colors used to present 3D text for selected items/cells/rows/columns.
<b>ForeColor</b>	Specifies text color for a cell.
<b>Format</b>	Specifies how non-string data are converted into strings.
<b>GradientColorFrom</b>	Specifies what color will be used to paint a gradient on the background.
<b>GradientColorTo</b>	Specifies what color will be used to paint a gradient on the background.
<b>GradientStyle</b>	Specifies the way a gradient will be drawn on the background.
<b>MarginLeft, MarginTop, MarginRight, MarginBottom</b>	Specify an offset between the cell boundaries and any text or graphics contained within a cell.
<b>MultiLine</b>	Specifies whether text is wrapped.
<b>Picture</b>	Specifies a picture to be drawn in a cell.
<b>PictureAlignment</b>	Specifies how a picture is aligned in a cell.
<b>PictureSelected</b>	Specifies a picture to be drawn in a cell when it is selected.
<b>PictureWidth, PictureHeight</b>	Determine the width and height of the picture displayed in a cell. These properties are measured in twips.
<b>RTFStyle</b>	Specifies if the text, as specified by the Text property should be interpreted and displayed as RTF formatted text and has to be shown this way.
<b>SelBackColor</b>	Specifies the background color in a selected cell.
<b>SelForeColor</b>	Specifies text color for a selected cell.
<b>Selectable</b>	Provides control over the end-user's ability to select specific List/Tree items, or Cells, Rows, Columns in a Grid.
<b>SelBorderColor</b>	Determines the default border color displayed around a cell when it becomes selected.
<b>SelBorderStyle</b>	Determines how borders will be drawn around a cell when it becomes selected.
<b>Tag</b>	Tag for a cell.
<b>Text</b>	Specifies the string which is displayed in a cell. Default property.
<b>TextAlignment</b>	Specifies how text is aligned in a cell.
<b>Url</b>	Specifies the URL which can be used if the

**WebAutoNavigate** property is enabled.

#### Supported TListCellDefObjects include:

Valid TListCellDef object references include:	Provides reference to:
TList1. <b>DefItemCellDef</b>	Default for all items in a simple List or Tree, and for 1 <sup>st</sup> column of items within a grid
TList1. <b>ItemCell</b> (Index)	Item in simple tree or list
TList1. <b>LevelDefs</b> (Level)	All items at a particular hierarchic level – only applies to 1 <sup>st</sup> column
TList1. <b>Grid.GridCellDef</b> TList1. <b>ItemGrid</b> (Index). <b>GridCellDef</b>	Default for all cells in a Grid
TList1. <b>Grid.Cells</b> (Row, Col). <b>CellDef</b> TList1. <b>ItemGrid</b> (Index). <b>Cells</b> (Row, Col). <b>CellDef</b>	a grid cell
TList1. <b>Grid.ColDefs</b> (Col). <b>CellDef</b> TList1. <b>ItemGrid</b> (Index). <b>Coldefs</b> (Col). <b>CellDef</b>	a grid column
TList1. <b>Grid.RowCellDef</b> (row) TList1. <b>ItemGrid</b> (Index). <b>RowCellDef</b> (row)	a grid row

#### Examples

```
' Set the default Background color for all items in a Grid
TList1.Grid.GridCellDef.BackColor = RGB ( 200, 0, 128)

' Set the Background color for a specific Column
Dim objTICellDef as TListCellDef
Set objTICellDef = TList1.Grid.Coldefs ( 4 ).CellDef
objTICellDef.BackColor = vbBlue

' Set default Editing Style for List or Tree cells to Checkbox
Set objTICellDef = TList1.DefItemCellDef.
objTICellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
objTICellDef.EditInfo.Checkbox.States = TLCHK_2STATES
```

## TListCheckBox Object

### Description

**TListCheckBox** object controls the editing / data entry behavior of grid or tree cells. The **TListCheckBox** provides support for both 2 and 3 state checkboxes (as determined by the **States** property). A two state checkbox is either **Checked** or **UnChecked**. A three state checkbox may also be in a **Grayed** (mixed) state. TList's default behavior is to use 2 state checkboxes. It is also possible to specify customized pictures for each state using the **CheckedPicture**, **UncheckedPicture**, and **GrayedPicture** properties. Each Checkbox state has an associated value as specified by the checkbox **CheckedValue**, **UnCheckedValue** and **GrayedValue** properties. By default these values are 0 - unchecked, 1 - checked, 2 – grayed (similar to VB checkbox control values).

### Properties:

Property	Description
<b>Appearance</b>	This property determines whether TList will show default images for checkboxes in 2-d or

	3-d presentation, or whether a custom image is being show
<b>CheckedPicture, UncheckedPicture and GrayedPicture</b>	The <b>CheckedPicture</b> , <b>UncheckedPicture</b> and <b>GrayedPicture</b> set alternative pictures for each checkbox state.
<b>CheckedValue, UncheckedValue and GrayedValue</b>	These properties determine the <b>Value</b> of the cell in which a checkbox appears according to the state of the checkbox.
<b>Options</b>	The set of flags for controlling the behavior of the CheckBox edit object.
<b>States</b>	This property defines the number of states (2 or 3) which the CheckBox can take. A two state checkbox is either Checked or UnChecked. A three state checkbox may also be in a Grayed (mixed) state.

### Syntax

[TListCellDef].EditInfo.**CheckBox**

### Data Type

TListCheckBox object

### Example

```
'Sample 1
'Use checkbox editing for all cells in column 4 of a grid
Dim tlCell as TListCellDef
Set tlCell = TList.Grid.Coldefs( 4 ).CellDef
tlCell.EditInfo.Style = TLEDITINFO_CHECKBOX
OR
TList.Grid.Coldefs(4).CellDef.EditInfo.Style = TLEDITINFO_CHECKBOX

'Sample 2
' Specify editing settings for checkboxes in Grid cells
' Use checkboxes for all items in Column 4 of the Grid
With TList1.Grid.Coldefs(4).CellDef
    ' Set Editing Style
    .EditInfo.Style = TLEDITINFO_CHECKBOX
    ' Set values for each possible checkbox state
    .EditInfo.CheckBox.States = TLCHK_3STATES
    .EditInfo.CheckBox.CheckedValue = "Passed"
    .EditInfo.CheckBox.UncheckedValue = "Failed"
    .EditInfo.CheckBox.GrayedValue = "Not Tested"
    .EditInfo.CheckBox.GrayedPicture = LoadPicture ("somepath\NotTested.bmp")
End With
'set the checked/unchecked state of the checkboxes in rows 10, 11 and 12
TList1.Grid.Cells(10,4).CheckBoxValue = "Passed" 'checks the checkbox
TList1.Grid.Cells(11,4).CheckBoxValue = "Failed" 'unchecks the checkbox
TList1.Grid.Cells(11,4).CheckBoxValue = "Not Tested" 'display NotTested bitmap
```

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object

---

## TListColDef Object

### Description

This object represents a specific column of the **TListGrid** object. It provides access to the grid owning the column, and to the **CellDef** object determining the appearance of cells in this column. It also allows hiding a column, moving a column, or setting the **ValueName** determining what data appears in the column.

### Properties

Property	Description
<b>ValueName</b>	Returns an attribute – determines which ItemValues appear in the column.
<b>CellDef</b>	Returns a <b>TListCellDef</b> object. Default.
<b>Grid</b>	Returns a reference to the Grid object which owns this <b>ColDef</b> .
<b>Index</b>	Index in the <b>ColDefs</b> Collection.
<b>Visible</b>	Specifies whether the corresponding column is visible on the screen or not.
<b>Selected</b>	Returns the status of the corresponding column.
<b>Width</b>	Specifies the width of a column in twips.

### Methods

Property	Description
<b>MoveTo</b>	Changes column's position.

### Examples

```
'Set the background color for all cells in the third column
TList1.Grid.ColDefs(2).CellDef.BackColor = RGB ( 200, 0, 128)
'...or
'hide fourth column
TList1.Grid.ColDefs(3).Visible = False
```

---

## TListColDefs Object Collection

### Description

The **TListColDefs** object is a standard collection object that holds a series of **TListColDef** objects, representing all columns of the **TListGrid** object.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in a collection.
<b>Item</b>	Id is an <b>ValueName</b> name or an ordinal number of the value in the <b>Values</b> collection. Default, Read-only

### Examples

```
'loop through all columns of the grid and display the visibility status
Dim iCnt As Long
For iCnt = 0 To TList1.Grid.ColDefs.Count
    Debug.Print "Column(" & iCnt & ") Visible=" TList1.Grid.ColDefs(iCnt).Visible
Next
```

---

## TListComboBox Object

### Description

The **TListComboBox** object is used for convenient data editing by selecting a value from a list. Its properties and methods manage the selection list and the editing window layout/presentation.

#### Properties:

Property	Description
<b>Style</b>	Defines the editing window style
<b>Options</b>	Set additional options
<b>Items</b>	References the collection of combobox list items - TListComboItem objects
<b>EditAreaMinHeight, EditAreaMaxHeight</b>	Defines the maximum height of the text entry area
<b>EditAreaMinWidth, EditAreaMaxWidth</b>	Defines the minimum height of the text entry area
<b>DropDownItemHeight</b>	Defines the item height in the drop down list
<b>DropDownMaxHeight</b>	Defines the maximum height of the list portion of the combobox
<b>DropDownWidth</b>	Defines the maximum width of the list portion of the combobox

#### Syntax

[TListCellDef].EditInfo.**ComboBox**

#### Data Type

**TListComboBox** object

#### Data Type

```
'set up combobox in-place editor for a grid column
TList.Grid.ColDefs(4).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
TList.Grid.ColDefs(4).CellDef.EditInfo.ComboBox.Items.Add ("Combo Item 1")
TList.Grid.ColDefs(4).CellDef.EditInfo.ComboBox.Items.Add ("Combo Item 2")
TList.Grid.ColDefs(4).CellDef.EditInfo.ComboBox.Items.Add ("Combo Item 3")
```

#### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListSpin object, TListDateTime object, TListComboltems object, TListComboltem object

---

## TListComboltem Object

#### Description

The **TListComboltem** object represents an item in the TListComboBox list (all items in the drop-down list). Its properties and methods determine the data and presentation of a single list box item .

#### Properties:

Property	Description
<b>CellValue</b>	This property is a unique key identifying the ComboBox list item. This value assigned to the cell after the corresponding list item is selected
<b>DisplayValue</b>	This property specifies the text shown for each item in in the drop down portion of a combobox during in-place editing. This value displayed in the cell after the corresponding list item is selected.

<b>DisplayPic</b>	This property determines the picture shown for a combobox list item in the drop down portion of the combobox. This picture displayed for unselected item
<b>PictureSelected</b>	This property determines the picture shown for a combobox list item in the drop down portion of the combobox. This picture displayed for selected item.
<b>BackColor, ForeColor</b>	Background and Text Colors of unselected item
<b>SelBackColor, SelForeColor</b>	Background and Text Colors of selected item
<b>PictureAlignment</b>	Specifies the picture alignment within a cell.
<b>TextAlignment</b>	Specifies the text alignment for a cell.
<b>Alignment</b>	Determines the alignment between text and picture in a cell.
<b>BorderStyle</b>	Determines what borders will be drawn around a cell.
<b>BorderColor</b>	Border color for a cell
<b>GradientStyle</b>	Sets the background style
<b>GradientColorFrom, GradientColorTo</b>	Set the colors used for gradient drawing
<b>Font</b>	Specifies the text font for a cell

### Syntax

[TListCellDef].EditInfo.ComboBox.Items.**Item**(index)

### Data Type

**TListComboItem** object

### Remarks

The **Items** property of a **TListComboBox** references a collection of **TListComboItem** objects. The **Item** property of the **Items** collection is an array property referencing a specific ComboBox List item – a **TListComboItem** object.

**TListComboItem** objects may be used for two different purposes:

- A) to control presentation of a list item in a TList editing **ComboBox** .
- B) to specify automatic "intelligent" formatting of tree or grid cells based on the value of the cell.

( See section "*Using the TListComboBox for Intelligent Formatting in " HOW TO USE TLIST In-Place Editing / Data Entry mechanisms"*")

The **CellValue** property of each **TListComboItem** object is a unique key identifying the **ComboBox** list item.. The value of this property is identical to the first parameter in the **Add** method of the **TListComboItems** object.

### Example

In the following example the TLComboItem.**CellValue** property is equal to "Root Item":

```
Dim TLComboItem as TListComboItem
TList.ItemCell(0).EditInfo.Style = TEDITINFO_COMBOBOX
Set TLComboItem = TList.ItemCell(0).EditInfo.ComboBox.Items.Add ("Root Item")
```

Upon selecting an item from the ComboBox, the value of the **TListCellDef** object is set to the value of the **CellValue** property of the selected ComboBox item.

In the next example TList will display the string "hundred" within the ComboBox instead of the number "100" in the first item after executing the assignment in the last line of code:

```
' Add an item to TList with text "100"
TList.AddItem "100"
' Set the EditStyle for this item to ComboBox
TList.ItemCell(0).EditInfo.Style = TL_Combobox
' Add a ComboBox Item with a value of "100" and a display value of "hundred"
Dim TListComboItem as TListComboItem
Set TListComboItem = TList.ItemCell(0).EditInfo.ComboBox.Items.Add ("100")
TListComboItem.DisplayValue = "hundred"
```

All the other properties of the **TListComboItem** can be used for "intelligent" formatting of the cell – similar to the way the word "hundred" is displayed in place of the value "100" in the example above.

The next example demonstrates how to set the background color to red for all grid cells containing the string "Apples":

```
'Set up a Tree item and an ItemGrid under it containing 6 rows
' and 5 columns with some data
TList1.AddItem "Root"
TList1.ItemGrid(0).Cols = 5
TList1.ItemGrid(0).Rows = 6
' Set text in grid cells to either "Apples" or "Usual"
For row = 1 To 5
  For col = 1 To 4
    TList1.ItemGrid(0).Cells(row, col).Value.Value =
      If( (row + col) Mod 2 = 1, "Apples", "Usual" )
  Next col
Next row

' Now use ComboBox to format cells
' Set the EditStyle for all cells in the ItemGrid to ComboBox
TList1.ItemGrid(0).GridCellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
' Add a ComboBox Item with a value of "Apples" and a background color of Red
TList1.ItemGrid(0).GridCellDef.EditInfo.ComboBox.Items.Add("Apples").BackColor = RGB(255, 0, 0)
```

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object, TListComboItems object

---

## TListComboItems Object

### Description

The **TListComboItems** object holds a collection of **TListComboItem** objects. It is used for manipulating the **Combobox** List.

### Properties:

Property	Description
<b>Count</b>	Returns the number of the items in an object collection.
<b>Sorting</b>	Sets the sorting order
<b>SortingKey</b>	Sets the sorting key



<b>SortingStyle</b>	Sets the sorting method
---------------------	-------------------------

### Methods:

Method	Description
<b>Add</b>	Adds a new item to the collection
<b>SafeAdd</b>	Adds a new item to the collection, or overwrites an existing item
<b>Item</b>	This property returns an object specified by the given index.
<b>Clear</b>	Removes all the items from the list
<b>RemoveItem</b>	Removes an item identified by its index
<b>Remove</b>	Removes an item identified by its value
<b>GetItemByCellValue</b>	Returns an index from the given value

### Example

```
Dim objItems as TListComboItems
Set objItems = TList1.ItemCellDef( itemindex ).EditInfo.ComboBox.Items
objItems.Add Value,, DisplayValue
```

### Remarks

Each item in the collection has a unique key. This key corresponds to the first parameter in the **Add** method of the **TListComboItems** object. This key also corresponds to the value which will be set into the cell by selecting an item from the **ComboBox** drop-down list. This key can be read using the **CellValue** property. For example:

```
TList1.ItemGrid(i).ColDefs(2).EditInfo.ComboBox.Items(1).CellValue
```

The default attributes used to present each drop-down list item during editing are inherited from the attributes of the cell being edited.

For example:

```
TList1.Grid.cells(2,3).celldef.BackColor = RGB (0, 255, 0)
```

sets green background color for a particular cell. This color will be used as the default for all items in the combobox, but can be overridden for any item by setting another value for a checkbox item

For example:

```
TListEditInfo].ComboBox.Items.Add 10,, "Ten"
[TListEditInfo].ComboBox.Items.Add 20,, "Twenty"
[TListEditInfo].ComboBox.Items.Add 30,, "Thirty"
[TListEditInfo].ComboBox.Items.Add 40,, "Forty"
'first specify green background for a particular cell
'will be taken as default for all items in combobox drop down.
TList1.Grid.cells(2,3).celldef.BackColor = RGB (0, 255, 0)
'specify red background for the item with index = 1 and blue for item 3
[TListEditInfo].ComboBox.Items.Item(1).BackColor = RGB (255, 0, 0)
[TListEditInfo].ComboBox.Items.Item(3).BackColor = RGB (0, 0, 255)
```

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object, TListComboItem object

---

## TListDataObject Object

### Description

This object is a container for data being transferred from a source to a TList. The data is stored in the format defined by the method using this object. This object is accessible only during OLE drag/drop operations inside OLEDragDrop event.

### Properties

Property	Description
<b>Files</b>	Returns a list of all filenames used by a TListDataObject.

### Methods

Property	Description
<b>GetData</b>	Returns data from the TListDataObject in a specified format.
<b>GetFormat</b>	Checks whether a TListDataObject has data in a required format.

### Example

```
'see VB sample 8 (Using OLE drag/drop) for details
Dim objItems as TListComboltems
Private Sub TList1_OLEDragDrop(ByVal data As TListDataObject, effect As Long, ByVal Button As Integer, _ ByVal Shift
As Integer, ByVal X As Double, ByVal Y As Double)
'See, what type of data the user drop onto TList
Select Case True
Case data.GetFormat(15)
'Don't allow the user to MOVE files from folders or desktop, just let to copy them
effect = 1 ' only copying
Dim I as Long , Files As TListDataObjectFiles
Set Files = data.Files
For I = 0 To Files.Count - 1
ProcessDroppedFile Files(I), Tlist1.DropTarget
Next
End Select
End Sub
```

---

## TListDataObjectFiles Object

### Description

The TListDataObjectFiles Object is a collection whose elements represent a list of all filenames used by a **TListDataObject** object (such as the names of files that a user drags to or from the Windows File Explorer.). This object is accessible only during OLE drag/drop operations inside OLEDragDrop event.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in a collection.
<b>Item</b>	Returns a specific member of a collection object by position.

### Example

```
'see sample 8 (Using OLE drag/drop) for details
Dim objItems as TListComboltems
Private Sub TList1_OLEDragDrop(ByVal data As TListDataObject, effect As Long, ByVal Button As Integer, _ ByVal Shift
As Integer, ByVal X As Double, ByVal Y As Double)
'See, what type of data the user drop onto TList
```

```

Select Case True
Case data.GetFormat(2)
...
Case data.GetFormat(3)
...
Case data.GetFormat(8)
...
Case data.GetFormat(15)
'We don't allow the user to MOVE files from folders or desktop, just let to copy them
effect = 1 ' only copying
Dim I as Long , Files As TListDataObjectFiles
Set Files = data.Files
For I = 0 To Files.Count - 1
    ProcessDroppedFile Files(I), Tlist1.DropTarget
Next
End Select
End Sub

```

## TListDateTime Object

### Description

**TListDateTime** object manages date / time style data editing.

The **TListDateTime** object provides support for setting the earliest and latest valid dates (**Min** and **Max** properties), date display format ( **Format** and **FormatString** properties), and the use of a drop down calendar for date selection (**Options** property).

### Properties:

Property	Description
<b>Format</b>	This property specifies the format of a string representing the date in the editing window.
<b>FormatString</b>	This property specifies a custom format for the string in the editing window. The property takes effect only when the <b>Format</b> property is set to TLFORMAT_CUSTOM.
<b>Min, Max</b>	These properties set the earliest and latest values for the DateTime edit object.
<b>Options</b>	The set of flags for controlling the behavior of the DateTime edit object.

### Syntax

[TListCellDef].EditInfo.**DateTime**

### Data Type

**TListDateTime** object

### Example

```

With Tlist1.Grid.ColDefs(3).CellDef
.EditInfo.Style = TLEDITINFO_DATE_TIME
.EditInfo.DateTime.format = TLFORMAT_LONG_DATE
.EditInfo.datetime.options = TLDATETIME_OPT_CALENDAR
'limit the date range that can be entered
.EditInfo.datetime.Min = cDate("2/01/01")
.EditInfo.datetime.Max = cDate("2/28/01")
End With

```

### See Also

---

## TListEditInfo Object

### Description

The **TListEditInfo** object provides convenient access to the in-place editing interface of the TList control. This object holds the editing style and provides access to the particular edit object used for in-place editing.

### Properties

Property	Description
<b>Style</b>	Sets the editing / presentation style for the specified TList data cell or cells
<b>Editable</b>	This property specifies determines whether a cell is editable. This property overrides EditingMode property settings and ItemEditText and CellEdit properties.
<b>Checkbox</b>	Returns a reference to a TList CheckBox editing object
<b>ComboBox</b>	Returns a reference to a TList ComboBox editing object
<b>DateTime</b>	Returns a reference to a TList DateTime editing object
EditableStartOptions	This property specifies additional options which determine whether a cell is editable. This property works in conjunction with Editable property.
KeyboardNavigateWhileEditing	Controls how TList reacts to navigation keys while the user is editing.
<b>Spin</b>	Returns a reference to a TList Spin editing object
<b>Textbox</b>	Returns a reference to a TList Textbox editing object

### Data Type

**TListEditInfo** object

### Remarks

To reset the Editing style to none – set the EditInfo property of the **TListCellDef** object to “Nothing”

Set TList1.Grid.Coldefs(1).CellDef.EditInfo = Nothing

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object

---

## TListEditingChangeInfo Object

### Description

The **TListEditingChangeInfo** object provides convenient access to the data the user enters using built-in editors (**TListTextBox**, **TListComboBox**, **TListCheckBox**, etc)

### Properties:

Property	Description
<b>ValueType</b>	Specifies the type of value being entered in the edit object.
<b>Value</b>	Specifies the new value of the edit object.
<b>OldValue</b>	Specifies the value of the edit object before it changed
<b>SelStart</b>	Specifies the starting point of text selected; indicates the position of the insertion point if no text is selected.
<b>SelLength</b>	Specifies the number of characters selected.
<b>OldSelStart, OldSelLength</b>	Returns the starting point and the number of characters selected; or returns the position of the insertion point if no text was selected.
<b>EditInfoObject</b>	Returns the reference to the <b>TListEditInfo</b> object being used for current editing. Note: Changes, to the TListEditInfo object within the GridCellEditingChange or ItemEditing Change events, will take affect only after the cell's editing is cancelled or completed.

### Data Type

**TListEditingChangeInfo** object

### Remarks

This object is presented as a parameter of the **GridCellEditingChange** and **ItemEditing Change** events.

### Example

```
Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _
    ByVal objGridCell As TListGridCell, _
    ByVal objChangeInfo As TListEditingChangeInfo)

    ' update a textbox on a form during editing
    ' as user moves through items in editing combobox

    If objChangeInfo.ValueType = TLED_CHANGE_COMBOBOX_LISTINDEX
        'get a reference to the combobox item that was selected
        Dim objComboltem as TListComboltem
        Set objComboltem =
            objChangeInfo.EditInfoObject.ComboBox.Items(objChangeInfo.Value)
        'update outside TextBox object to current combobox list selection
        Text1.Text = objComboltem.CellValue
        Text1.BackColor = objComboltem.BackColor
    End If
End Sub
```

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object

GridCellEditingChange and ItemEditingChange events

---

## TListGrid Object

### Description

The **TListGrid** object provides convenient access to the data of a specific grid object created inside TList control.

It can represent both a grid with hierarchical tree in the one column and a grid as a children of the tree items.

Here are properties, methods, functions and events of TListGrid object:

### Properties

<b>Property</b>	<b>Description</b>
<b>ActivationMode</b>	Specifies the activation/navigation mode that TList uses when within a Grid.
<b>ActiveCell</b>	Returns a reference to active TListGridCell object.
<b>ActiveRow</b>	Returns a reference to the active TListGridRow object.
<b>AllowResizing</b>	Returns or sets a value that determines whether the user is allowed to resize rows and columns.
<b>AutoFillRowTitles</b>	Determines what default text will be shown in row titles.
<b>AutoFillColTitles</b>	Determines what default text will be shown in column titles.
<b>AutoSizeOptions</b>	Specifies whether TList will automatically resize grid columns to fit the contained data.
<b>BackColorBkg</b>	Determines the background color of the control in areas not filled by a Grid.
<b>BorderStyle</b>	Determines the border style of grid objects.
<b>CellEdit</b>	Enables, concludes or cancels an in-place editing operation.
<b>Cells</b>	Returns a TlistGridCell object for the specified cell.
<b>Col and Row</b>	Returns or sets the coordinates of the active cell in a grid.
<b>ColDefs</b>	Returns a TlistColDefs object collection, which has a number of TlistColDef objects storing column formatting.
<b>Cols and Rows</b>	Returns or sets the total number of columns or rows in a TList.
<b>DragColumnsEnabled</b>	Enables or disables the end-user ability to drag columns of a grid with the mouse.
<b>GridCellDef</b>	Returns a TlistCellDef object, which specifies default property settings for each cell in a grid.
<b>ColTitleCellDef</b>	Returns a TlistCellDef object, which specifies default property settings for each column header in a grid.
<b>RowTitleCellDef</b>	Returns a TlistCellDef object, which specifies default property settings for each row header in a grid. These are default settings for all <b>Cells</b> (XX, 0) cells.
<b>GridLinesColor</b>	Specifies the color in which grid lines are drawn.
<b>GridLinesStyle</b>	Returns or sets a value that determines grid line type.

GridLines3DStyle	Returns or sets a value that determines 3D grid line style (two-color or three-color lines).
GridLines3DLightColor	Specifies the main color used for drawing 3D grid lines.
<b>GridLines3DShadowColor</b>	Specifies the shadow color used for drawing 3D grid lines.
<b>HasCell</b>	Determines whether a specified cell exists.
<b>MouseCol</b>	Returns the current mouse position in row and column coordinates.
<b>MouseRow</b>	
NonScrollableColumns	Specifies the number of columns on the Left which should remain frozen in a Grid when scrolling horizontally.
<b>ParentItemIndex</b>	ParentItemIndex is -1 if a grid is a Tree Grid, otherwise this grid is an Item Grid and ParentItemIndex returns index of the item which owns the grid.
<b>RowCellDef</b>	Specifies default settings for the whole row.
<b>RowDefs</b>	Returns a reference to a <b>TListRowDef</b> object by specified index.
<b>RowHeight</b>	Returns or sets the height of the specified row.
<b>SelectedCells</b>	Returns a reference to a collection of TListGridCell objects that are currently selected.
<b>SelectedColumns</b>	Returns a reference to a collection of TListColDef objects that are currently selected.
<b>SelectedRows</b>	Returns a reference to a collection of TListRowDef objects that are currently selected.
<b>SelectionMode</b>	Controls the how objects (rows and/or cells) are selected within a particular TListGrid object.
<b>SelectionOptions</b>	Specifies some options that allow user to control the selection process more thoroughly.
<b>Sorted</b>	Initiates, halts or resets the sorting.
<b>SortingMethod</b>	Specifies ascending or descending sort order.
<b>SortingKey</b>	Specifies which column(s) should be used as the key for the sorting.
<b>SortingStyle</b>	Provides additional control such as numeric or case sensitive sorting
<b>ShowColTitles</b>	Determines whether column titles are visible.
<b>ShowRowTitles</b>	Determines whether row titles are visible.
ToolTip	Specifies a tooltip object to be applied for the whole grid.
<b>TreeGrid</b>	True if this is Tree Grid and False otherwise.
<b>Visible</b>	True if Grid is visible, False otherwise.

## Methods

Property	Description
<b>Activate</b>	Activates any cell / row of a TListGrid object.
<b>AddRow</b>	Adds a row to a Tlist control.
<b>AutoSizeRow</b>	Resets the height of a specified row to fit the maximum height of the data in the row, and returns the new row height in pixels.
<b>AutoSizeColumn</b>	Resets the width of a specified column to fit the maximum width of the data in the column and returns the new column width in pixels.
<b>RemoveRow</b>	Removes a row from a TList control.
<b>ItemIndexToRow</b>	Converts the index of an item into corresponding Grid row index.
<b>RowToItemIndex</b>	Converts from a row number within a TList Grid object to an index value for TList.

### Example

```
Private Sub Form_Load()
    TList1.Redraw = False
    TList1.Grid.Cols = 3
    TList1.Grid.Rows = 6

    TList1.Grid.Cells(0, 1).Value = "Country"
    TList1.Grid.Cells(0, 1).CellDef.TextAlignment = TLTEXTALIGNMENT_LEFT_CENTER
    TList1.Grid.Cells(0, 2).Value = "Flag"

    TList1.Grid.Cells(1, 2).CellDef.Picture = picCanada.Picture
    TList1.Grid.Cells(1, 1).Value = "Canada"
    TList1.Grid.Cells(2, 2).CellDef.Picture = picItaly.Picture
    TList1.Grid.Cells(2, 1).Value = "Italy"
    TList1.Grid.Cells(3, 2).CellDef.Picture = picJapan.Picture
    TList1.Grid.Cells(3, 1).Value = "Japan"
    TList1.Grid.Cells(4, 2).CellDef.Picture = picSpain.Picture
    TList1.Grid.Cells(4, 1).Value = "Spain"
    TList1.Grid.Cells(5, 2).CellDef.Picture = picGreatBritain.Picture
    TList1.Grid.Cells(5, 1).Value = "Great Britain"

    TList1.Grid.ShowRowTitles = False
    TList1.Grid.ColDefs(2).CellDef.TextAlignment = TLTEXTALIGNMENT_NOT_VISIBLE
    TList1.Grid.ColDefs(1).CellDef.PictureAlignment = TLPICTUREALIGNMENT_NOT_VISIBLE
    TList1.Grid.GridLinesStyle = 0 'NONE

    TList1.Redraw = True
End Sub
```

---

## TListGridCell Object

### Description

This object defines data and formatting style for a cell of a grid. This object may be returned as a parameter of the events generated while an end-user interacts with a grid object. This object allows you to retrieve the TListGrid object that this cell belongs to and also get information about the row and column indexes.

### Properties

Property	Description
----------	-------------



<b>CellDef</b>	Returns TListCellDef object, which specifies graphic attributes for the cell.
<b>CheckboxValue</b>	Returns or sets a value that determines the state of a checkbox in a grid cell.
<b>Grid</b>	Returns the TListGrid object which owns a specified cell.
<b>Row</b>	Returns the row index of a cell.
<b>Col</b>	Returns the column index of a cell.
<b>Value</b>	Returns a reference to TListValue object, which stores data shown in the cell. Default.
<b>Top, Left</b>	Determines the position of the cell in TList (in twips). These properties return valid values only for visible cells.
<b>Height, Width</b>	Determines the size of the cell (in twips).
<b>Selected</b>	Returns the status of the corresponding element (cell, row or column).
<b>ToolTip</b>	Specifies a tooltip object to be applied to a grid cell.

### Example

```
Private Sub TList1_GridCellClick(ByVal GridCell As TListGridCell, ByVal Button As Integer)
' Change data shown in column 2 in response to click in Row0 (title row), Column 2.
If GridCell.Row = 0 And GridCell.Col = 2 Then
' user clicked in the caption cell of the third column, let's select/deselect the whole column
If GridCell.Grid.ColDefs(GridCell.Col).CellDef.BackColor = RGB(0, 0, 255) Then
GridCell.Grid.ColDefs(GridCell.Col).CellDef.BackColor = &H1 'remove the back color
Else
GridCell.Grid.ColDefs(GridCell.Col).CellDef.BackColor = RGB(0, 0, 255)
End If
End If
End Sub
```

---

## TListLevelDef Object

### Description

This object is used to specify formatting and other settings for a particular level of the tree. Using this object you can specify the settings for all items at some particular hierarchic level in one step without walking through the tree and applying the settings item by item manually.

### Properties

Property	Description
<b>CellDef</b>	Returns a TListCellDef object, which specifies default graphic attributes for all items of the specified level. Default.
<b>Indentation</b>	Returns the indentation level.
<b>PictureClosed</b>	Specifies a picture for collapsed items.
<b>PictureLeaf</b>	Specifies a picture for items which do not have children.
<b>PictureOpen</b>	Specifies a picture for expanded items.
<b>Sorted</b>	Initiates, halts or resets the sorting.
<b>SortingMethod</b>	Specifies ascending or descending sort order.

<b>SortingKey</b>	Specifies what data (visible text or hidden ItemValues) should be used as the key for the sorting.
<b>SortingStyle</b>	Provides additional control such as numeric or case sensitive sorting

### Example

```
'following line of the code will change the back color for all items on the 0 level
TList1.LevelDefs(0).CellDef.ForeColor = RGB(128, 0, 0)

'Without LevelDefs you have to use following code
Dim iCnt as Long
For iCnt = 0 To TList1.ListCount-1
  If TList1.Indent(iCnt) = 0 Then
    TList1.ItemBackColor(iCnt) = RGB(128, 0, 0)
  End If
Next
```

---

## TListLevelDefs Object Collection

### Description

The **TListLevelDefs** object is a standard collection that holds a series of TListLevelDefs objects. A **LevelDef** object contains formatting information for a particular level of the tree.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in a collection.
<b>Item</b>	Id is an ordinal number of the LevelDef object in the LevelDefs collection. Default, Read-only

### Example

```
'applying the sorting for all items in the tree
Dim objLevel as TListLevelDef
For Each objLevel In TList1.LevelDefs
  ObjLevel.SortingStyle = TL_B_SORT_IGNORE_CASE
  ObjLevel.Sorted = True
Next
```

---

## TListNode Object

### Description

A **TListNode Object** provides simple and convenient access to all properties describing and controlling an item. Using **TListNode** interface it is possible to get a reference to the parent item, next and previous siblings to add a new subordinate item.

### Properties

Property	Description
<b>Add</b>	Adds new node(item) using specified type of relationship with this node
<b>BackColor</b>	Specifies the background color for this node
<b>CellDef</b>	Returns reference to the TListCellDef object describing appearance of internal item cell

<b>Children</b>	Returns reference to a TListNodes collection containing the children of an item
<b>ChildrenCount</b>	Returns the number of children
<b>EnumIndex</b>	Returns TList index (zero-based enumeration of all items in TList disregarding their visibility).
<b>Expand</b>	Specifies whether a node is expanded
<b>FirstSibling</b>	Returns a reference to the first sibling of this node
<b>Grid</b>	Returns a reference to the node grid object
<b>Image</b>	Specifies a picture to be displayed with this node
<b>Indent</b>	Specifies the hierarchic indentation level of this node
<b>Index</b>	Returns the index of this node in the parent list
<b>LastSibling</b>	Returns a reference to the last sibling of this node
<b>Next</b>	Returns a reference to the next sibling of this node
<b>Parent</b>	Returns or sets the parent object of this node
<b>Prev</b>	Returns a reference to the previous sibling of this node
<b>SelBackColor</b>	Specifies the selected background color for this node
<b>Selected</b>	Determines whether this node is selected. (valid only when MultiSelection mode turned ON)
<b>SelectedImage</b>	Specifies a selected picture to be displayed with this node
<b>Sorted</b>	Initiates, halts or resets the sorting.
<b>SortingMethod</b>	Specifies ascending or descending sort order.
<b>SortingKey</b>	Specifies what data (visible text or hidden ItemValues) should be used as the key for the sorting.
<b>SortingStyle</b>	Provides additional control such as numeric or case sensitive sorting
<b>Tag</b>	Specifies a hidden data associated with this node
<b>Text</b>	Specifies a node's text
<b>Values</b>	Returns reference to collection of data associated with this node
<b>VirtualCount</b>	Specifies the number of virtual children for this node
<b>VirtualParent</b>	Specifies whether children of this node are virtual or not
<b>Visible</b>	Returns the visibility state of this node (read-only)

## Example

```
Sub AddSubItems(ParentNode As TListNode)
' Add child nodes to whatever node is passed to this routine
Dim CurNode As TListNode
Dim Index As Long
For Index=1 To 5
    Set CurNode = ParentNode.Add("Child" & Str(Index), TLNODERELATION_CHILD)
    CurNode.Image = LoadPicture()
    CurNode.SelectedImage = LoadPicture()
    CurNode.BackColor = RGB(0, 255, 0)
    CurNode.Tag = Index
Next Index
End Sub
```

---

## TListNodes Object Collection

### Description

The **TListNodes Object** is a standard collection that holds a series of **TListNode Objects**. **TListNodes** collection represents all subordinates items of the parent node (item). Using **TListNodes** interface a developer can easily add, remove and modify tree structure without referencing TList itself. This allows manipulating the tree structure in an object-oriented way

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in a collection.
<b>Item</b>	Returns reference to the item specified by index. Default, Read-only
<b>Add</b>	Adds new item to the collection.
<b>Remove</b>	Removes specified item from the collection.
<b>Clear</b>	Clears contents of collection.

### Example

```
'here is a sample where you can pass a reference to the some specified TListNodes Object inside the procedure and
manipulate with it without accessing the TList object.
Sub AddRootItems(RootNodes As TListNodes)
    Dim CurNode As TListNode
    Dim Index As Long
    For Index=1 To 10
        ' The Add method of a TListNode Object returns a reference to the newly added node
        Set CurNode = RootNodes.Add("Root"& Str(Index), -1, TLNODERELATION_CHILD)
        ' Pass the new root as a node to a subroutine which adds child nodes
        Call AddSubItems(CurNode)
    Next Index
End Sub
```

---

## TListReport Object

### Description

This object provides user with printing interface that allows to have full control over the printed report structure.

### Properties

Name	Description
<b>AbortWindowStyle</b> <b>, AbortWindow</b>	Specifies how printing progress will be displayed to the user
<b>AutoNewPage</b>	Specifies if TList should automatically start a new page when there is a need. Otherwise it is up to the programmer to specify a new page (for example, using Printer object's NewPage method).
<b>FirstItem</b> , <b>LastItem</b>	Specifies the range of items to be printed
<b>LeftMargin</b> , <b>TopMargin</b> , <b>RightMargin</b> <b>BottomMargin</b>	Specify the region on a page where TList will print data. These properties are measured in twips.
<b>Pages</b>	Returns the collection of pages, prepared for printing
<b>PostScriptDC</b>	Should be set to True in case of printing to a postscript printer
<b>PreviewMode</b>	Specifies if TList should print on a printer or a screen DC (usually used to make a preview)
<b>PreviewPageWidth</b> <b>PreviewPageHeight</b>	Specify the width and the height of a page to be printed (measured in twips)
<b>PrintBackground</b>	Should be set to True in order to force TList to print control's background
<b>PrinterObject</b>	Specifies destination of printed output - the default Printer object, a PictureBox control, or a DC handle.
<b>PrintLevels</b>	Specifies how many levels of the Tree will be printed.
<b>ShowColumnTitles</b>	Specifies if Tree Grid headers should be repeated on each page.
<b>Zoom</b>	Specifies a Zoom factor in percents (from 1% to 10000%)

### Methods

Name	Description
<b>Start</b>	Initiates the printing process
<b>PrepareForPrinting</b>	Creates and formats a collection of pages to be printed
<b>PrintingStop</b>	Stops the printing process
<b>IsPrinting</b>	Returns the status of the current printing process

### Remarks

For most applications it is simpler to use the PrintOneStep method than to use the Report Object. The Report object should only be used for more complex printing which the PrintOneStep Method can not satisfy .

## Example

```
Private Sub Command1_Click()
    On Error GoTo Printer_error
    Printer.Print "" ' initializes printing

    Set TList1.Report.PrinterObject = Printer
    TList1.Report.FirstItem = 0
    TList1.Report.LastItem = -1 'whole tree
    TList1.Report.Zoom = 100 '100% zoom factor
    TList1.Report.LeftMargin = 200
    TList1.Report.TopMargin = 200
    TList1.Report.RightMargin = 200
    TList1.Report.BottomMargin = 200
    TList1.Report.AbortWindowStyle = 1 'default window
    TList1.Report.PrintBackground = True 'print background picture

    'prepare for printing, creating the collection of pages
    Dim PageCount As Long
    PageCount = TList1.Report.PrepareForPrinting
    'start printing process
    TList1.Report.Start 0, PageCount - 1
    'end up printing process
    Printer.EndDoc
Exit Sub
Printer_error:
    MsgBox Err.Description, vbOKOnly Or vbExclamation
    ' do nothing in this case
End Sub
```

---

## TListPage Object

### Description

This object is used to provide all the information about the current page during printing process. Using properties of this object you can get the current page number (pages are numbered starting with 1), specify margins for the page, or get the index of the first and the last items printed on the page.

### Properties

Name	Description
<b>PageNumber</b>	Returns the current page number .
<b>LeftMargin</b>	Specifies the left margin for the page
<b>TopMargin</b>	Specifies the top margin for the page
<b>RightMargin</b>	Specifies the right margin for the page
<b>BottomMargin</b>	Specifies the bottom margin for the page
<b>FirstItem</b>	Returns the index of the first item on the page.
<b>LastItem</b>	Returns the index of the last item on the page.. <b>Note</b> the LastItem property returns -1 in the BeginPage event and a valid value in the EndPage event.

### Example

```

'printing headers
Private Sub TList1_BeginPage(ByVal PrinterObject As Variant, ByVal CurrPage As TListPage)
    PrinterObject.CurrentX = TList1.Report.LeftMargin
    PrinterObject.CurrentY = 0
    PrinterObject.Print "Page (" & CurrPage.PageNumber & ")"

    PrinterObject.Line (TList1.Report.LeftMargin, TList1.Report.TopMargin)- _
        (PrinterObject.ScaleWidth - TList1.Report.RightMargin, TList1.Report.TopMargin), RGB(0, 0, 0)
End If
End Sub

```

---

## TListPages Object Collection

### Description

This object is a collection of **TListPage** objects. This object is automatically created by **TListReport** object in answer to the **PrepareForPrinting** method call.

### Properties

Name	Description
<b>Count</b>	Returns the number of pages, read only
<b>Item</b>	Returns a page object, read only

### Example

```

Private Sub Command1_Click()
    On Error GoTo Printer_error
    Printer.Print "" ' initializes printing

    Set TList1.Report.PrinterObject = Printer
    TList1.Report.FirstItem = 0
    TList1.Report.LastItem = -1 'whole tree

    'prepare for printing, creating the collection of pages
    Dim PageCount As Long
    PageCount = TList1.Report.PrepareForPrinting
    'enumerate the TListPages collection in order to display the indexes of
    'the first and last items on the page
    Dim objPage as TListPage
    For Each objPage In TList1.Report.Pages
        Debug.Print "Page=" & objPage.PageNumber _
            & " First=" & objPage.FirstItem _
            & " LastItem=" & objPage.LastItem
    Next
    'do the printing after...
    Printer.EndDoc
Exit Sub
Printer_error:
    MsgBox Err.Description, vbOKOnly Or vbExclamation
    ' do nothing in this case
End Sub

```

---

## TListSpin Object

### Description

**TListSpin** object is used for managing increment/decrement style data editing.

The **TListSpin** object displays spin buttons during editing allowing the user to increment or decrement the value by a discrete amount (**Step** property) when clicking on the spin buttons. The **Max** and **Min** properties may be used to set a range of values. The **Options** property may be used to specify placement of increment/decrement buttons, response to arrow keys and the wrapping of the range.

#### Properties:

Property	Description
<b>Min, Max</b>	These properties set the maximum and minimum values for the <b>TListSpin</b> edit object.
<b>Options</b>	The <b>Options</b> property is a bit flag property, each bit is a flags specifying the presentation and behavior of the <b>TListSpin</b> edit object.
<b>Step</b>	This property sets the <b>TListSpin</b> edit object increment value - the amount by which the value of the cell is changed in response to the user clicking the spin buttons.

#### Syntax

[TListCellDef].EditInfo.**Spin**

#### Data Type

TListSpin object

#### Example

```
With TList1.Grid.ColDefs(3).CellDef
.EditInfo.Spin.Min = 0
.EditInfo.Spin.Max = 100
.EditInfo.Spin.Step = 10
.EditInfo.Spin.Options = TLSPIN_OPT_WRAP _
                        Or TLSPIN_OPT_ARROWKEYS _
                        Or TLSPIN_OPT_HORZ
End With
```

#### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListTextBox object, TListCheckBox object, TListComboBox object, TListDateTime object

---

## TListTextBox Object

#### Description

The **TListTextBox** object is used for managing textbox style data editing. Its properties and methods manage the the editing window layout/presentation. The **TListTextBox** provides support for setting a Maximum data entry length (see **MaxLength** property), a Minimum and Maximum (see **MinHeight**, **MaxHeight**, and **HeightScale** properties), a minimum and maximum width (see **MinWidth**, **MaxWidth** properties) and automatic adjustment of editing box size (see **Options** property)

#### Properties:

Property	Description
<b>HeightScale</b>	This property defines measurement units for the <b>MaxHeight</b> and <b>MinHeight</b> properties.
<b>MinHeight, MaxHeight</b>	These properties set the minimum and



	maximum vertical size of the Edit window in pixels
<b>MaxLength</b>	This property sets the maximum number of characters supported for data entry (typing) within a data cell.
<b>MinWidth, MaxWidth</b>	These properties set the maximum and minimum size of the TextBox window in pixels.
<b>Options</b>	This property applied to a <b>TListTextBox</b> object determines whether the text-editing window for the associated cell or cells is automatically resized as the end-user edits the text.

### Syntax

[TListCellDef].EditInfo.**TextBox**

### Data Type

**TListTextBox** object

### Example

```
With TList1.Grid.ColDefs(2).CellDef
.EditInfo.Style = TLEDITINFO_TEXTBOX
.EditInfo.TextBox.Options = TLTEXTBOX_OPT_AUTOWIDTH
.EditInfo.TextBox.MinWidth = 150      'min width of edit window in pixels
.EditInfo.TextBox.MaxWidth = 250     'max width of edit window in pixels
.EditInfo.TextBox.MaxLength = 10     'max number of characters in edit window
End With
```

### See Also

How to Use TList In-Place Editing/Data Entry

TListEditInfo object, TListEditingChangeInfo object, TListCheckBox object, TListComboBox object, TListSpin object, TListDateTime object

---

## TListValue Object

### Description

This object provides an interface to both the hidden and visible data of the item or grid cell.

### Properties

Property	Description
<b>Value</b>	Stores data themselves.
<b>ValueName</b>	Returns corresponding attribute.
<b>ItemIndex</b>	Returns the index of the item which owns this value.

### Example

```
'Create a tree and associate hidden data with each item
'Note the use of the special index value "-3" which always refers to the most recently added item.
TList1.AddItem "Item 1", -1
TList1.ItemValues(-3, "HiddenData1").Value = "Some Data 1"
TList1.ItemValues(-3, "HiddenData2").Value = 52
TList1.AddItem "Item 2", -1
TList1.ItemValues(-3, "HiddenData1").Value = "Some Data 2"
TList1.ItemValues(-3, "HiddenData2").Value = 152
...
```

---

## TListValues Object Collection

### Description

The TListValues collection is a standard collection object that holds a series of TListValue objects, which offer an interface to both the hidden and visible data of the item or grid cell.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in a collection.
<b>Item</b>	Id is an ordinal number of the value in the Values collection.

### Example

```
'Create a tree with associated hidden data for each item
'Note the use of the special index value "-3" which always refers to the most recently added item.
TList1.AddItem "Item 1", -1
TList1.ItemValues(-3, "HiddenData1").Value = "Some Data 1"
TList1.ItemValues(-3, "HiddenData2").Value = 52
TList1.AddItem "Item 2", -1
TList1.ItemValues(-3, "HiddenData1").Value = "Some Data 2"
TList1.ItemValues(-3, "HiddenData2").Value = 152
...
'enumerating all hidden values for the first item via TListValues collection
Dim objValue as TListValue
For Each objValue In TList1.ItemValues(0)
    Debug.Print "Item(0) ValueName=" & objValue.ValueName & " ValueData=" & objValue.Value
Next
```

---

## TListSelectedGridCells Object

### Description

The **TListSelectedGridCells** object is a collection that holds a series of **TListGridCell** objects representing all selected cells in the active grid. The collection items are indexed from according to their position, from Left to Right, Top to Bottom. The collection is returned by the **SelectedCells** property of the **TListGrid** object.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in the collection - ie: the number of selected cells.
<b>Item</b>	Returns a reference to the item specified by index in the collection. Read-only
<b>Add</b>	Adds new cell to the collection. This is the same as selecting corresponding cell of the active grid object.
<b>Remove</b>	Removes cell from the collection. This is the same as deselecting corresponding cell of the active grid object.
<b>Clear</b>	Clears contents of collection. Deselects all cells of the active grid object.

<b>Grid</b>	Returns the Grid object this collection belongs to.
-------------	---

### Example

```
Dim objSelectedCells as TListSelectedGridCells
Set objSelectedCells = TList1.Grid.SelectedCells
Dim objCell as TListGridCell
Debug.Print "Selected Cells"
For Each objCell In objSelectedCells
    Debug.Print "objCell =" & objCell.Row & "," & objCell.Col & ")"
Next
```

## TListSelectedGridColumnns Object

### Description

The **TListSelectedGridColumnns** object is a collection that holds a series of **TListColDef** objects representing all fully selected columns (where all cells in the column are selected) in the active grid. The collection items are indexed in sequence from the left most selected column to the right most selected column. The collection is returned by the **SelectedColumns** property of the **TListGrid** object.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in the collection - ie: the number of selected columns.
<b>Item</b>	Returns a reference to the column specified by index in the collection. Read-only
<b>Add</b>	Adds new column to the collection. This is the same as selecting corresponding column of the active grid object.
<b>Remove</b>	Removes column from the collection. This is the same as deselecting corresponding column of the active grid object.
<b>Clear</b>	Clears contents of collection. Deselects all columns of the active grid object.
<b>Grid</b>	Returns the Grid object this collection belongs to.

### Example

```
Dim objSelectedColumns as TListSelectedGridColumnns
Set objSelectedColumns = TList1.Grid.SelectedColumns
Dim objColDef as TListColDef
Debug.Print "Selected Columns"
For Each objColDef In objSelectedColumns
    Debug.Print "ColDef Index =" & objColDef.Index & objColDef.ValueName
Next
```

## TListSelectedGridRows Object

### Description

The **TListSelectedGridRows** object is a collection that holds a series of **TListRowDef** objects representing all fully selected rows (where all cells in the row are selected) in the active grid. The collection items are indexed in sequence from the top to the bottom. The collection is returned by the **SelectedRows** property of the **TListGrid** object.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in the collection - i.e.: the number of selected rows.
<b>Item</b>	Returns a reference to the row specified by index in the collection. Read-only
<b>Add</b>	Adds new row/item to the collection. This is the same as selecting corresponding row of the active grid object.
<b>Remove</b>	Removes row from the collection. This is the same as deselecting corresponding row of the active grid object.
<b>Clear</b>	Clears contents of collection. Deselects all rows of the active grid object.
<b>Grid</b>	Returns the Grid object this collection belongs to.

### Example

```
Dim objSelectedRows as TListSelectedGridRows
Set objSelectedRows = TList1.Grid.SelectedRows
Dim objRowDef as TListRowDef
Debug.Print "Selected Rows"
For Each objRowDef In objSelectedRows
    Debug.Print "Row Index =" & objRowDef.Index
Next
```

---

## TListRowDefs Object

### Description

The **TListRowDefs** object is a collection that holds a series of **TListRowDef** objects representing all rows in the grid. The collection items are indexed in sequence from the top to the bottom. The collection is returned by the **RowDefs** property of the **TListGrid** object.

### Properties

Property	Description
<b>Count</b>	Returns the number of objects in the collection - i.e.: the number of selected rows.
<b>Item</b>	Returns a reference to the row specified by index in the collection. Read-only

### Example

```
Dim objRows as TListRowDefs
Set objRows = TList1.Grid.RowDefs
Dim objRowDef as TListRowDef
Debug.Print "All Grid Rows"
For Each objRowDef In objRows
    Debug.Print "Row Index =" & objRowDef.Index
Next
```

---

## TListRowDef Object

### Description

The **TListRowDef** object represents a row in a **TListGrid** object. This object provides accesses to get/set information specific for a particular row of the grid object.

### Properties

Property	Description
<b>RowIndex</b>	Returns the index of this object in the grid. Note: rows in grid are numbered from 0. Read-only.
<b>CellDef</b>	Returns a reference to the <b>TListCellDef</b> object that describes the format and edit settings for this row. Read-only.
<b>Values</b>	Returns a reference to the <b>TListItemValues</b> object that holds named data associated with a TList item. Note: this is the same data that user can get using TList1.ItemValues() property call specifying as a parameter the index of the item corresponding this row object. Read-only.
<b>ItemIndex</b>	Returns the index of the item that corresponds this row of the grid object. Note: It is the same as using RowToItemIndex call specifying the row index as a parameter. Read-only
<b>Selected</b>	Specifies whether the grid row is selected.
<b>Grid</b>	Returns the Grid object this row belongs to.

### Example

```
'deselect all rows in the grid
Call TList1.Grid.SelectedRows.Clear()
'select first, third and fifth rows in the grid
TList1.Grid.RowDef(1).Selected = True
TList1.Grid.RowDef(3).Selected = True
TList1.Grid.RowDef(5).Selected = True
```

## TVWSelectedNodes Object Collection (TListTreeView)

### Description

The TVwSelectedNodes object is a standard collection that holds a series of Node objects. TVwSelectedNodes collection represents all selected nodes in the current tree. The collection items are indexed in sequence from the first selected node to the last selected node. In single select mode the SelectedNodes collection can only contain one object. The collection is returned by the SelectedNodes property of the TListTreeView control.

### Properties

Property	Description
Count	Returns the number of objects in the Selected nodes collection - ie: the number of selected nodes.
Item	Returns a reference to the item specified by index in selected nodes collection. Read-only
Add	Adds new item to the collection. This is the same as selecting corresponding node of the tree.
Remove	Removes item from the selected nodes collection. Deselects the node.
Clear	Clears contents of collection. Deselects all nodes of the tree.

### Example

```
Dim objSelectedNodes as TVwSelectedNodes
Set objSelectedNodes = TListTreeView.SelectedNodes
Dim objNode as Node
Debug.Print "Selected Nodes"
For Each objNode In objSelectedNodes
```

```
Debug.Print "Node Index=" & objNode.IndexNext
```

---

## TListPoint Object

### Description

A **TListPoint** object defines a point location with X, Y Values

### Properties

The **TListPoint** object has the following properties:

Property	Description
X	This property specifies X coordinate.
Y	This property specifies Y coordinate.

### Example

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
With objTooltipArgs
.Window.LocalPosition.Left = .MouseLocal.X
.Window.LocalPosition.Top = .MouseLocal.Y
.Window.RecalculateSize False
End With
End Sub
```

### See Also

TListToolTipWindow object, TListRectangle object

---

## TListRectangle Object

### Description

The **TListRectangle** object has the following properties and methods.

### Properties

The **TListRectangle** object has the following properties:

Property	Description
Left	This property specifies left coordinate of the rectangular area.
Top	This property specifies top coordinate of the rectangular area.
Right	This property specifies right coordinate of the rectangular area.
Bottom	This property specifies bottom coordinate of the rectangular area.

### Example

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
With objTooltipArgs
.Window.LocalPosition.Left = .MouseLocal.X
.Window.LocalPosition.Top = .MouseLocal.Y
.Window.RecalculateSize False
End With
End Sub
```

## See Also

TListToolTipWindow object, TListPoint object

---

# TListShowTooltipArgs Object

## Description

The TListShowTooltipArgs object defines the content and presentation of a tooltip. This object is presented as a parameter of the ShowTooltip event, giving the programmer full control over tooltip appearance and behavior.

## Properties

The TListShowTooltipArgs object has the following properties:

Property	Description
Cancel	Set this property to true in order to prevent the tooltip window from being displayed.
Tooltip	Returns a <b>TListTooltip</b> object specifying the content of the tooltip window to be displayed
ItemIndex	Returns the item index of the item currently located under the mouse.
GridCell	If mouse is over a grid cell, this property returns an instance of a TListGridCell object. Otherwise this property returns nothing.
MouseScreen	Returns the column index of a cell.
MouseLocal	Returns a <b>TListPoint</b> object with read only properties that identify the current mouse position relative to the top left corner of the TList control, measured in pixels
Window	Returns a <b>TListTooltipWindow</b> object

## Example

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    If .GridCell Is Nothing Then
      .Tooltip.Text = "this is not a grid cell"
    Else
      .Tooltip.Text = "this is a grid cell"
    End With
  End Sub
```

## See Also

TListToolTipWindow object, TListToolTip object

---

# TListTooltip Object

## Description

A TListTooltip object defines the text, picture, and presentation properties of the tooltip window.

## Properties

The **TListTooltip** object has the following properties:

Property	Description
Text	Defines the text displayed in the tooltip window If it is necessary to force line breaks, use embedded carriage returns / linefeed

characters within the text string, and make sure that the ToolTipStyle.MultiLine property is set to True.

**Picture**

Defines the picture displayed in a tooltip window.

**Style**

Property returns a **TListTooltipStyle** object, specifying the colors and text alignment used for display of the tooltip.

**Example**

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    .ToolTip.Text = "This is line 1" & vbCrLf & "This is line 2"
    .ToolTip.Style.BorderStyle = TLBORD_2PIXELBORDER
    .ToolTip.Style.MultiLine = TL_MULTILINE_TRUE
  End With
End Sub
```

**See Also**

ItemToolTip property, ToolTip property, ToolTip **property of** TListShowTooltipArgs parameter of ShowToolTip event

---

## TListTooltipStyle Object

**Description**

A **TListTooltipStyle** object defines the colors and text alignment used for display of a tooltip.

**Properties**

TListTooltipStyle object has the following properties:

Property	Description
BackColor	Specifies the background color of the tooltip window.
ForeColor	Specifies the text color of the tooltip window.
PictureAlignment	Specifies the picture alignment of the tooltip window.
TextAlignment	Specifies the text alignment of the tooltip window.
BorderStyle	Specifies the border style of the tooltip window.
BorderColor	Specifies the border color of the tooltip window.
MultiLine	Specifies whether text is wrapped inside tooltip window.

**Example**

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    .ToolTip.Style.BorderStyle = TLBORD_2PIXELBORDER
    .ToolTip.Style.MultiLine = TL_MULTILINE_TRUE
  End With
End Sub
```

**Applies to**

**ToolTip.Style** property of the TListShowTooltipArgs parameter of ShowToolTip event



---

# TListTooltipWindow Object

## Description

A **TListTooltipWindow** object defines the location and size of a ToolTip Window.

## Properties and Methods

TListTooltipStyle object has the following properties:

Property	Description
LocalPosition	Returns a <b>TListRectangle</b> object with properties: Left, Top, Right and Bottom, which allow the programmer to get or set the position and size of the ToolTip window – measured in pixels relative to the top left corner of the TList control.
ScreenPosition	Returns a <b>TListRectangle</b> object with properties: Left, Top, Right and Bottom, which allow the programmer to get or set the position and size of the ToolTip window – measured in pixels relative to the top left corner of the screen.

TListTooltipStyle object has the following method:

Method	Description
<b>RecalculateSize</b> (bKeepWidth as Boolean)	<p>Calling this method <b>AFTER</b> setting the Text, Picture, and Style properties, instructs TList to recalculate the required window size for the specified tooltip contents.</p> <p>Call this method with parameter TRUE to instruct TList to use the maximum possible width of the text for the item/cell under the cursor ( generally this is the screen width )</p> <p>Call this method with parameter FALSE to instruct TList to calculate the width for the tooltip window based on the tooltip text and picture size</p> <p>In either case ( True or False ) the tooltip height will be calculated to fit the text which is wrapped within the calculated width</p> <p>Note – this method call will reset the height and possibly even the width of the tooltip window even if they were explicitly assigned earlier via LocalPosition and ScreenPosition properties.</p>

## Example

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    .Window.LocalPosition.Left = .MouseLocal.X
    .Window.LocalPosition.Top = .MouseLocal.Y
    .Window.RecalculateSize False
  End With
End Sub
```

## Applies to

**Window** property of the TListShowTooltipArgs parameter of ShowToolTip event



# Properties, events, methods, functions reference

---

## AbortWindowStyle and AbortWindow Properties

### Applies To

TListReport object

### Description

The **AbortWindowStyle** property specifies how printing progress will be displayed to the user:

The **AbortWindow** property specifies the window handle of a dialog to be displayed instead of the standard printing progress dialog.

Not available at design-time.

### Syntax

```
TListReportObject.AbortWindowStyle [= enum%]  
TListReportObject.AbortWindow [= WindowHandle&]  
[form.]TList.Report.AbortWindowStyle [= enum%]  
[form.]TList.Report.AbortWindow [= WindowHandle&]
```

### Settings

The **AbortWindowStyle** property settings are:

Setting	Description
0 (None)	No visual feedback is shown as TList prints
1 (Internal window)	TList creates and shows the standard printing progress dialog, the same window can be used to cancel printing
2 (user-defined window)	In this case the <b>AbortWindow</b> property must be set to a valid window handle. TList will use the specified window to display the printing progress.

### Data Type

Long

---

## About Property

### Applies to

TList object

### Description

At design time you can double-click this property in the property window to popup the About box which contains copyright information, a date stamp and the version of the control. The **About** property is not available at runtime.

### See Also

Version property

---

## Activatable Property (TListCellDef Object)

### Applies to

TListCellDef object

### Description

This property provides control over the end-user's ability to navigate to any particular item/cell/row using keyboard, mouse or using properties (for example via **Activate** method for TListGrid object and **ListIndex** property for the tree). Cells which can not be activated can not receive the focus and will be skipped over when navigating by keyboard and ignored when clicked by mouse.

This property is helpful for specifying non-accessible regions in the grid or tree.

### Syntax

[TListCellDef]. <b>Activatable</b> = [enum]
---

### Settings

Settings for the **Activatable** property are:

Constant	Setting	Description
TL_ACTIV_INHERITED ( Default )	-1	The ability to activate the object is inherited. A cell's ability to be activated may be inherited from the Row, the Column, the Grid or the entire TList object.  <i><b>Note</b> – this value is Read-only. Reading the Activatable property always returns either 0 or 1</i>
TL_ACTIV_DISABLED	0	The specified cell(s) can not be activated ( can not receive focus). Such cells will be skipped over when navigating by keyboard)
TL_ACTIV_ENABLE	1	The specified cell can be activated regardless of the Activatable property setting for any higher level object to which it belongs. For instance; setting Activatable = TL_ACTIVE_ENABLE for a cell will allow that cell to be activated even if the settings for the parent Row, Column or entire Grid are TL_ACTIV_DISABLED

### Default Value

By default this property is set to **TL\_ACTIV\_INHERITED** for all TListCellDef objects.

### Example

```
'Prevent any cell in the 3rd column from being activated
TList1.Grid.ColDefs(3).CellDef.Activatable = TL_ACTIV_DISABLED
' ...
'Prevent the Grid Cell in Row 2, Column 4 from being Activated
TList1.Grid.Cells(2,4).CellDef.Activatable = TL_ACTIV_DISABLED
' ...
'make the item with index 100 read-only for navigation
```

```
TList1.ItemCell(100).Activatable = TL_ACTIV_DISABLED
```

### Remarks

Use the **Selectable** property to disable selecting of Activatable elements.

For single-selection mode (**MultiSelect** property = 0) setting **Activatable** property to **False** disables the ability to change the selection as well.

Items, rows, cells, marked as unable to be activated are also prevented from become active as a result of program code. For example, setting ItemCell(11).Activatable = TL\_ACTIV\_DISABLED prevents TList.ListIndex =11 from having effect.

---

## Activate Method (TListGrid object)

### Applies to

TListGrid object

### Description

This method activates any cell / row of a TListGrid object (unless such activation disabled by the cell or row's Activatable property).

Visually the Active Cell/Row is shown with a Focus Rectangle (depending on the setting of the **FocusRectStyle** property)

### Syntax

```
[TListGrid].Activate (ByValRowIndex As Long, ByVal ColIndex As Long)
```

### Settings

Valid values forRowIndex and ColIndex parameters are dependant upon the ActivationMode property as presented in the following table:

ActivationMode property	ValidRowIndex values	ValidColIndex values
TL_ACTIVMODE_ITEM, Or TL_ACTIVMODE_ROW	Any valid row index for the particular <b>TListGrid</b> object.  Setting to -1 will deactivate any active row, removing the focus (as if setting TList1.ListIndex = -1).	-1  ( ColIndex can't be set to anything except -1 in this Activation Mode)
TL_ACTIVMODE_CELL	Any valid row index.  Setting to -1 will deactivate any active cell regardless of ColIndex value.	Any valid column index  Setting to -1 will deactivate any active cell regardless orRowIndex value.
TL_ACTIVMODE_MIXED	Any valid row index.  Setting to -1 will deactivate any active row / cell.	Any valid column index.  Setting to -1 will deactivate any active cell and make the row with index =RowIndex the active one

---

**Note:** setting both ColIndex andRowIndex parameters to -1 is a valid operation for any ActivationMode setting that will deactivate any previously active cell / row / item.  
Calling the Activate method with valid Row and Column Index parameters, while in Mixed

Activation Mode automatically toggles TList to cell-by-cell activation state within mixed mode as if user hits <Shift> <Space>.

---

### Remarks

Calling this method to activate a cell or row of a grid, that was not previously active, forces the Grid containing the newly activated cell or row to become active.

### Example

```
Make row 5 of the grid active
' Assumes item or Row Mode Activation Mode
Call TList1.Grid.Activate(5, -1)
' ...
'Activate Cell in Row 10, Column 11
' Assumes Cell or Mixed Activation Mode
Call TList1.Grid.Activate(10,11)
' ...
'deactivate the currently active cell
Call TList1.Grid.Activate(-1,-1)
```

### See Also

Activable property

---

## ActivationMode Property (TListGrid object)

### Applies to

TListGrid object

### Description

This property specifies the activation / navigation mode that TList uses when within a Grid.

- By **Activation** we mean – placing the focus upon a row or cell – generally resulting in the drawing of a focus rectangle around the cell.
- By **Navigation** we mean the moving around within TList – changing the activated row or cell by way of keyboard or mouse.
- By **Selection** we mean highlighting a row or cell – generally resulting in reverse coloring of the cell. Activation and Navigation do NOT necessarily imply Selection. **TIP:** If desired, cells may be easily selected and deselected during activation and deactivation within the GridCellActivate and GridCellDeactivate Events.

Outside a Grid (when dealing with ordinary List or Tree items in TList ) the user navigates on a row by row basis – clicking anywhere in a row, or using the keyboard arrow keys.

Within a Grid there are four types of navigation / selection: item-by-item, row-by-row, cell-by-cell and row/cell combined mode.

- **Item mode** (default) – This mode primarily designed to provide backward compatibility with older versions of TList.  
In Item mode activation within a grid is handled as if the rows are just ordinary items within the main list / tree; The focus rectangle includes the entire item area including the row title. Clicking the mouse anywhere along a row – even beyond the left and right boundaries of the grid – can activate the row  
We recommend setting **SelectionMode** = TL\_SELMODE\_INHERITED in combination with this mode.
- **Row mode** - is the same as Item mode except
  - a) Mouse clicks adjacent to grid rows but outside grid boundaries do not activate any grid rows..
  - b) The focus rectangle is drawn around row cells excluding row title cell.

- c) Grid Column Titles of ItemGrids are not drawn as selected when the user selects the parent of an ItemGrid, and
- d) Row Titles are not drawn as selected unless the  
TL\_SELOPT\_MARKSELECTED\_ROWCOLTITLE flag is set for the grid's  
SelectionOptions property..
- **Cell mode** – In Cell mode TList allows activation and selection of individual grid cells, navigating from cell to cell and selecting cells by either keyboard or mouse. Mouse Clicks adjacent to grid cells but outside the grid boundaries do not activate any grid cells..
- **Mixed mode** – In Mixed mode the user may toggle between Row Mode behavior and Cell Mode behavior, full row and cell-by-cell, using either keyboard or mouse action. Clicking the mouse on any row title switches TList to full row selection / activation mode. Clicking on any particular grid cell switches to cell by cell selection / activation. The user also toggle the selection / activation mode using the <SHIFT>+ <SPACE> keyboard combination. The drawing of a focus rectangle around either a single cell or a complete row, as well as the selection of a single cell or complete row, will continue in the same state as the user navigates through the tree until the user toggles, the selection / activation mode or the mode is changed by code. Activation (responding to the mouse input) affects only internal grid area, clicking outside grid ignored.

### Modes comparison table

This is a list of differences in the behavior of the control between modes.

End User Action	Response: ActivationMode = Item Mode	Response: ActivationMode = Row, Cell, or Combined Mode
<b>Clicking outside ItemGrid object (on the left or right side)</b>	The item/row to the left or right of the click location is activated	The Activation of items, cells, and rows within TList is NOT changed. It is, as the click action did not occur.
<b>Clicking item grid column titles (for tree item that has an ItemGrid as a child).</b>	Activates parent item.	Parent item is not activated
<b>Clicking a cell, or using the keyboard to navigate</b>	The focus rectangle is drawn around the entire item area (from left side of TList's client area to the right side and with the height of the row)	The focus rectangle is drawn around either the row or an individual cell depending on the Activation mode.

### Syntax

[TListGrid].ActivationMode = [enum]

### Settings

Settings for the **ActivationMode** property are:

Constant	Setting	Description
TL_ACTIVMODE_ITEM	0	(Default). Activation on item-by-item basis.
TL_ACTIVMODE_ROW	1	Activation/selection on row-by-row basis.
TL_ACTIVMODE_CELL	2	Activation/selection on cell-by-cell basis
TL_ACTIVMODE_MIXED	3	Combination of two modes (row and cell) with ability to switch between them via keyboard or mouse.

## [See Also](#)

**KeyboardActivation** property

---

# ActiveGrid Property

## [Applies to](#)

TList object

## [Description](#)

Returns a reference to a TListGrid object whose cell was clicked.

Read-only at run time.

## [Syntax](#)

```
TList1.ActiveGrid
```

## [Example](#)

```
MsgBox "The active Grid has : " _  
    & Str$(TList1.ActiveGrid.Rows) _& " Rows and " _  
    & Str$(TList1.ActiveGrid.Cols) & " Columns"
```

---

# ActiveCell Property (TListGrid object)

## [Applies to](#)

TListGrid object

## [Description](#)

Returns a reference to active TListGridCell object.

## [Syntax](#)

```
[TListGridCell] = [TListGrid].ActiveCell
```

## [Remarks](#)

This is a Read-Only property. The **Activate** method of TListGrid object should be used in order to activate any particular cell/row in some grid.

There will be no ActiveCell in the following cases

- TLGridObject.ActivationMode = TL\_ACTIVEMODE\_ROW

- TLGridObject.GridCellDef.Activation = TL\_ACTIV\_DISABLED

- (or TList does not have items - only captions)

- Before any cell is activated

- After TList1.ListIndex = -1

- After item that contains active cell was deleted

## [Example](#)

```
n'get an active cell and display it value  
Dim objCell as TListGridCell  
Set objCell = TList1.Grid.ActiveCell  
Debug.Print "objCell Value =" & objCell.Value.Value  
  
' Test whether any ActiveCell exists in main grid  
If TList.Grid.ActiveCell is Nothing Then ....  
  
' Test whether any ActiveCell exists in a specified TreeGrid  
If TList.ItemGrid(30).ActiveCell is Nothing Then ....
```



---

## ActiveRow Property (TListGrid object)

### Applies to

TListGrid object

### Description

Returns a reference to the active TListGridRow object.

### Syntax

```
[TListGridRow] = [TListGrid].ActiveRow
```

### Example

```
'get an active row and display it index  
Dim objRow as TListGridRow  
Set objRow = TList1.Grid.ActiveRow  
Debug.Print "objRow Index =" & objRow.Index
```

### Remarks

This is a Read-Only property - The **Activate** method of TListGrid object should be used in order to activate any particular cell/row in some grid.

This property always points to the active row even if the grid is in cell-by-cell activation mode (**ActivationMode** = TL\_ACTIVMODE\_CELL) and only the cell is visually displayed as active (marked by a focus rectangle).

---

## Add and SafeAdd Methods (TListComboItems Object)

### Applies to

TListEditInfo object

### Description

These methods add new items to the ComboBox drop-down list.

### Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.Add CellValue [, Picture] [, DisplayValue]  
[TListEditInfo].EditInfo.ComboBox.Items.SafeAdd CellValue [, Picture] [, DisplayValue]
```

### Values

The method parameters are:

Parameter	Description
<b>CellValue</b>	Required. Values for a ComboBox Item. This value is the unique key for the collection
<b>Picture</b>	<b>(Optional)</b> . A picture corresponding to the <b>CellValue</b>
<b>DisplayValue</b>	<b>(Optional)</b> . This value will be displayed instead of the <b>CellValue</b> (see the <b>TListComboItem</b> object)

### Details

The **Add** method may be used to add new items to the combobox list. Attempting to add an item with a value already in the collection will trigger an error.

The **SafeAdd** method may be used either to add new items, OR to replace existing items. Using **SafeAdd** with a **CellValue** held by one of the existing items updates the **Picture** and **DisplayValue** for that item.

Both methods return a reference to the **TListComboItem** object.

## Example

```
With TList.Grid.Coldefs(1).CellDef
'align all pictures above the text
.Alignment = TLALIGNMENT_PICTURE_ABOVE_TEXT
With .EditInfo.ComboBox
'add four items to the combo list with pictures
.Items.Add 10, Picture1.Picture, "Ten"
.Items.Add 20, Picture2.Picture, "Twenty"
.Items.Add 30, Picture3.Picture, "Thirty"
.Items.Add 40, Picture4.Picture, "Fourty"
'Replace the list box item "Fourty" with "Forty"
.Items.SafeAdd 40, Picture4.Picture, "Forty"
'add another item to the combo list, but with text above picture
Dim objComboltem as TListComboltem
Set objComboltem = .Items.Add(50, Picture5.Picture, "Fifty")
objComboltem.Alignment = TLALIGNMENT_PICTURE_BELOW_TEXT
End With
End With
```

---

## Add Method (TListSelectedGridCells, TListSelectedGridColumnns and TListSelectedGridRows Objects)

### Applies to

**TListGrid** object

### Description

This method add new object to the a collection that holds a series of **TListGridCell**, **TListColDef** and **TListRowDef** objects representing all selected cells/columns or rows. Adding an object to one of these collections selects the correspondent visible element in the control.

### Syntax

```
[TListGrid].SelectedCells.Add(ByVal IRow As Long, ByVal vCol As Variant)
[TListGrid].SelectedColumns.Add(ByVal vCol As Variant)
[TListGrid].SelectedRows.Add(ByVal IRow As Long)
```

### Values

The method's parameters are:

Parameter	Description
<b>IRow</b>	Row index of the object to be selected.
<b>vCol</b>	Column index or column's value name of the object to be selected.

### Details

Attempting to add an object using incorrect row or column indexes will trigger an error.

### Example

a) Adding two cells to the collection

```
'select two cells (2, 2) and (3, 3) in the grid
Dim objSelectedCells as TListSelectedGridCells
Set objSelectedCells = TList1.Grid.SelectedCells
Call objSelectedCells.Add(2, 2)
Call objSelectedCells.Add(3, 3)
```

```
'display results of the selection
Dim objCell as TListGridCell
Debug.Print "Selected Cells"
For Each objCell In objSelectedCells
    Debug.Print "objCell =(" & objCell.Row & "," & objCell.Col & ")"
Next
```

## b) Adding two rows to the collection

```
'select two rows 3 and 5 in the grid
Dim objSelectedRows as TListSelectedGridRows
Set objSelectedRows = TList1.Grid.SelectedRows
Call objSelectedRows.Add(3)
Call objSelectedRows.Add(5)
'display results of the selection
Dim objRowDef as TListRowDef
Debug.Print "Selected Rows"
For Each objRowDef In objSelectedRows
    Debug.Print "Row Index =" & objRowDef.Index
Next
```

## c) Adding two columns to the collection

```
'select two columns 2 and 3 in the grid
Dim objSelectedColumns as TListSelectedGridColumn
Set objSelectedColumns = TList1.Grid.SelectedColumns
Call objSelectedColumns.Add(2)
Call objSelectedColumns.Add(3)
'display results of the selection
Dim objColDef as TListColDef
Debug.Print "Selected Columns"
For Each objColDef In objSelectedColumns
    Debug.Print "ColDef Index =" & objColDef.Index & objColDef.ValueName
Next
```

---

# Add Property

## Applies to

TList object

## Description

Adds items referenced by a *bookmark* or stored in a *tree buffer* as subordinates of the item specified by an index.

Not available at design time and write-only at run time.

## Syntax

```
[form.]TList.Add(index&) = tree_buffer&
or
[form.]TList.Add(index&) = bookmark&
```

## Remarks

All characteristics of a bookmarked item are maintained when it is added. Any children of the bookmarked item are also added and their parent/child relationships are maintained.

Use the **FreeBuffer** method to free memory after you are done with the *tree buffer*.

Setting the **Add** property updates a visible control unless the **Redraw** property is set to False.

Use an index of (-1) to copy into an empty TList. If TList is not empty and the **CurrentIndexMethod** property is set to 0 or 1, then an index of -1 adds all items from *tree buffer* to the end of the list. If TList is not empty and the **CurrentIndexMethod** is set to 2, then an index of -1 adds all items from *tree buffer* to the end of the list of subordinates' of the current parent as specified by **CurrentParent** property.

Note the **Add** property depends on the **VisualRoot** property settings.

### Data Type

Long

---

## Add Method (TListNode/TlistNodes objects)

### Applies to

TListNode object

### Description

Adds a new item to the tree and returns the corresponding **TListNode Object**.

### Syntax

```
Nodes.Add(ByVal Text As String, ByVal Target As Variant, ByVal Relationship As Integer)
Node.Add(ByVal Text As String, ByVal Relationship As Integer)
```

### Settings

The **Add** method has following parameters:

Parameters	Description
Text	Text for the new item.
Target	Index of a pre-existing Node object. The relationship between the new node and this pre-existing node should be specified by the next argument, Relationship.
Relationship	Specifies the relative placement of the new Node object, as described in Settings.

The settings for Relationship parameter are:

Constant	Value	Description
TLNODERELATION_FIRST	0	First. The Node is placed before all other nodes at the same level of the node specified by the Target parameter.
TLNODERELATION_LAST	1	Last. The Node is placed after all other nodes at the same level of the node specified by the Target parameter.
TLNODERELATION_NEXT	2	Next. The Node is placed after the node specified by the Target parameter.
TLNODERELATION_PREVIOUS	3	Previous. The Node is placed before the node specified by the Target parameter
TLNODERELATION_CHILD	4	Child. The Node becomes a child node of the node specified by the Target parameter.

### Remarks

The **Add** method returns a reference to the newly created **TListNode** object, so it is convenient to use this reference for changing some properties.

### Example

```
' add a last child to the 4th root in TList (remember TList starts counting at 0)
```

```
TList.Root.Add("New item", 3, TLNODERELATION_CHILD )
```

```
' add a new node after the 4rd root in TList and then set the color of the new node  
Dim Tnode as TListNode  
Set Tnode = TList.Root.Add("New item", 3, TLNODERELATION_NEXT )  
Tnode.BackColor = vbBlue
```

---

## Add Method (TVWSelectedNodes Object)

### Description

Selects an existing node of the tree.

### Syntax

```
TListTreeView.SelectedNodes.Add(objNode As Node)
```

### Settings

The **Add** method has one parameter:

Parameters	Description
ObjNode	A reference to the node object to be selected .

### Remarks

This method automatically updates the SelectedNodes collection.

```
'You can also select nodes using Selected property of the Node object  
([form.]TListTreeView.Nodes(Index).Selected=True).  
'This is equivalent to  
TListTreeView.SelectedNodes.Add( TListTreeView.Nodes(Index) )
```

In single select mode the SelectedNodes collection can only contain one object. If the **SelectionMode** property is set to 0 (single selection mode) calling the Add method will deselect any previously selected node and will update the SelectedItem property to point to the newly selected node.

### Example

```
' Select two nodes (first and third ones) using Add method of TVwSelectedNodes collection  
Dim objSelectedNodes as TVwSelectedNodes  
Set objSelectedNodes = TListTreeView.SelectedNodes  
objSelectedNodes.Add TListTreeView.Nodes( 1 )  
objSelectedNodes.Add TListTreeView.Nodes( 1 )
```

---

## AddAfter Method

### Applies to

**TList** object

### Description

This method inserts a new item immediately after an item specified by its index. It is a complementary method for **InsertItem**.

### Syntax

```
TList.AddAfter(ByVal Text As String, ByVal DestIndex As Long)
```

### Settings

The Text parameter specifies the text to display in the newly added item.

The DestIndex parameter specifies the index of the item after which the new item is added.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Remarks

The index of the item added with the **AddAfter** method will be one greater than the index given by the **DestIndex** parameter.

Setting of **AddAfter** method updates the control unless the **Redraw** property is set to **False**.

Note the **AddAfter** property depends on the **CurrentIndexMethod** as is the case for any other method where an index is specified.

## Example

```
Starting with
    Item 0
    Item 1
    Item 2

The call
TList1.AddAfter "new text", 1
results in
    Item 0
    Item 1
    new text
    Item 2
```

---

# AddItem Method

## Applies to

**TList** object

## Description

Adds a new item to the **TList** control at run time.

## Syntax

```
[form.]TList.AddItem item [,index&]
```

## Remarks

The **AddItem** method can be used to add items as new children of an existing item or as peers of an existing item. This behavior is determined by the setting of the **MSOutlineAdd** property.

**Case: MSOutlineAdd = True:** **TList** implements the **AddItem** method in the same manner as the **MSOutline** control included with Visual Basic:

- if an **index&** parameter is specified with the **AddItem** method, **TList** inserts the new item at that position. The new item index is therefore **index&**, and the index of all items later in the list are incremented by 1.  
The hierarchic indentation level of the new item depends on whether an item previously existed at the specified index. If the item is inserted before another item (the item at position **index&** before the add), the new item is inserted into the list using the existing item's indentation level (ie: as a peer). However, if the new item is added as the last item in the list (or the last subordinate of the **CurrentParent** if **CurrentIndexMethod** = 2), it will be inserted with the indentation level of 0 (or at the same level as the **CurrentParent** if **CurrentIndexMethod**=2).
- If **index&** is not specified in the **AddItem** method, the currently selected item determines where the new item is added. For example, if the **ListIndex** property is set to 2, the new item is added to the end of the subordinate items for the item whose **ListIndex** value is 2. In the case where **ListIndex** is set to -1 (no item selected), the item is added to the end of the list with an indentation level of 0.

Case: **MSOutlineAdd = False (default)**, TList implements the **AddItem** method as in earlier versions of TList:

- if an **index&** parameter is specified in the **AddItem** method, TList will interpret the second parameter in the **AddItem** method as the index of a parent to which the new item should be added as a subordinate.
- If no **index&** parameter is specified TList adds the new item to the end of the list (as defined by the **CurrentIndexMethod** property).

The **AddItem** method may also be used to add collimated data to a Tree. Set the **ConvertTabsToCols** property to True (default) and use the character specified in the **ColDelimiter** property to delimit columns when calling the **AddItem** method. When adding column data, **AddItem** works as follows (we assume that there are **ColDelimiter** characters in the input string, otherwise **ConvertTabsToCols** has no effect on the **AddItem** method):

- if the TList.**HasGrid** property is set to True (Tree Grid exists), **AddItem** will add columns or data only to the Tree Grid.
- if the TList.**HasGrid** property is set to False (Tree Grid doesn't exist), and **AddItem** is called to add an item with indentation zero – a Tree Grid is created and the required number of columns are added.
- if the TList.**HasGrid** property is set to False (Tree Grid doesn't exist), and **AddItem** is called to add an item with indentation bigger than 0 – Item Grid is created and the required number of columns are added.

If **MSOutlineAdd** is True the **VisualRoot** property doesn't effect the **AddItem** method behavior. If the **CurrentIndexMethod** property is 2, the **VisualRoot** property settings are taken into account.

### Example

Both of the following statements add an item to the list. In each case the text shown is "Item1". All other properties of this item will have the default settings.

```
TList1.AddItem "Item1"  
TList1.AddItem "Item1", 5
```

In the first case, Item1 is added either to the end of the list if **MSOutlineAdd** = False or as a child of the item specified by the **ListIndex** property if **MSOutlineAdd** = True.

The second statement adds an item as a child of the item referred to by index 5 if **MSOutlineAdd** = False, or immediately after the item referred to by index 5 if **MSOutlineAdd** = True.

The next example adds two items with four columns of data to a list, creating the columns if they do not already exist.

```
TList1.ConvertTabsToCols = True  
TList1.ColDelimiter = Asc("^")  
TList1.AddItem "Fred^Jones^SomeStreet^New York"  
TList1.AddItem "Sarah^Lenore^Another Ave^Florida"
```

### Notes

Improved performance may be obtained by use of the **Redraw** property, the *FastItemAdd* method, or TList File I/O to save and load complete trees in a single step.

Calls to this method update the control, unless the **Redraw** property is set to False.

Newly added items will not necessarily be visible depending on whether their parent is in an expanded state.

---

Note The **CurrentIndexMethod** property is used to interpret the indexing scheme.

---

### See Also

**InsertItem**, **ConvertTabsToCols**, **ColDelimiter** properties

---

## AddItem2 and AddItem2Ex Methods

### Applies to

TList object

### Description

These two methods may be used to create and add a new item to the end of the tree and simultaneously set certain formatting attributes.

Not available at design time.

### Syntax

```
[form.]TList.AddItem2 (ByVal Shift As Integer,  
    ➡ByVal Text As String,  
    ➡ByVal Options As Long) As Integer
```

```
[form.]TList.AddItem2Ex (ByVal Shift As Integer,  
    ➡ByVal Text As String,  
    ➡ByVal Options As Long  
    ➡ByVal FontName As String,  
    ➡ByVal FontSize As Integer,  
    ➡ByVal BackColor As Long,  
    ➡ByVal ForeColor As Long) As Integer
```

### Parameters

The function parameters are:

Constant	Description
TltTree	Name of the TList control.
Shift	Indentation for the new added item.
Text	Text for the new item.
Options	This parameter determines whether FontName, FontSize, BackColor, or ForeColor parameters are used. It also determines some additional settings for the new item. See Options parameter settings description below.
FontName	Name of the font used with the new item. This parameter is used if the Options parameter has the TLFAST_FONTNAME flag set.
FontSize	Size of the font used with the new item. This parameter is used if the Options parameter has the TLFAST_FONTSIZE flag set.
BackColor	Background color for the new item. This parameter is used if the Options parameter has the TLFAST_BACKCOLOR flag set.
ForeColor	Text color for the new item. This parameter is used if the Options parameter has the TLFAST_FORECOLOR flag set.

Flags used with *Options* parameter are:

Constant	Value	Description
TLFAST_BACKCOLOR	&H1	If this flag set, the <i>BackColor</i> parameter has valid data for new item's background color. Otherwise the <i>BackColor</i>



		parameter is not used.
TLFAST_FORECOLOR	&H2	If this flag set, the <i>ForeColor</i> parameter has valid data for new item's text color. Otherwise the <i>ForeColor</i> parameter is not used.
TLFAST_FONTNAME	&H4	If this flag set, the <i>FontName</i> parameter has valid data for new item's font name. Otherwise <i>FontName</i> parameter is not used.
TLFAST_FONTSIZE	&H8	If this flag set, the <i>FontSize</i> parameter has valid data for new item's font size. Otherwise the <i>FontSize</i> parameter is not used.
TLFAST_FONTITALIC	&H10	If this flag set, the new item's font will have its <i>Italic</i> style set regardless of the <b>FontItalic</b> property.
TLFAST_FONTBOLD	&H20	If this flag set, the new item's font will have its <b>Bold</b> style set regardless of the <b>FontBold</b> property.
TLFAST_FONTUNDER	&H40	If this flag set, the new item's font will have its <u>Underline</u> style set regardless of the FontUnder property.
TLFAST_FONTSTRIKE	&H80	If this flag set, the new item's font will have its <del>StrikeThrough</del> style set regardless of the FontStrike property.
TLFAST_NOTFONTITALIC	&H10 0	If this flag set, the new item's font will <b>NOT</b> have its <i>Italic</i> style set regardless of the <b>FontItalic</b> property.
TLFAST_NOTFONTBOLD	&H20 0	If this flag set, the new item's font will <b>NOT</b> have <b>Bold</b> style set regardless of the <b>FontBold</b> property.
TLFAST_NOTFONTUNDER	&H40 0	If this flag set, the new item's font will <b>NOT</b> have its <u>Underline</u> style set regardless of the FontUnder property.
TLFAST_NOTFONTSTRIKE	&H80 0	If this flag set, the new item's font will <b>NOT</b> have its <del>StrikeThrough</del> style set regardless of the FontStrike property.
TLFAST_EXPANDED	&H10 00	If this flag set, the new item's children will be expanded by default.
TLFAST_NOTEXPANDED	&H20 00	If this flag set, the new item's children will be collapsed by default.

### Returns

These functions return 0 if successful and an error code otherwise.

### Remarks

If the FontName parameter is not specified, the AddItem2Ex method will use TList's default font (as set by TList.FontName).

---

## AddRow Method

### Applies To

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference • 185

## TListGrid Object

### Description

Adds a row to a Grid object.

### Syntax

```
TListGridObject.AddRow (item As String [, ByVal GridRowNum As Variant] )
```

### Remarks

The **AddRow** method syntax has these parts:

Part	Description
<i>Item</i>	Required. A string expression displayed in the newly added row. To add multiple strings (for multiple columns in the row), use the delimiter specified by <b>ColDelimiter</b> property.
<i>GridRowNum</i>	Optional. A Long representing the position within the control where the new row is placed. For the first row, <i>GridRowNum</i> = 0. If <i>GridRowNum</i> is omitted, the new row is added at the end of the list at the same indentation level.

### Example

This example uses the **AddRow** method to add 100 items to a Grid. To try this example, paste the code into the Declarations section of a form with a **TList** control named TList1, and then press F5 and click on the form.

```
Private Sub Form_Click ()
    'Declare variables.
    Dim Entry As String
    Dim i As Integer
    'add 100 items to your TList.
    TList1.Grid.Cols = 2 ' Two cols per row.
    For i = 1 To 100 ' 100 entries.
        ' Create entry, use tab as column delimiter
        Entry = "Row Number " & Chr(9) & i
        TList1.Grid.AddRow Entry ' Add entry.
    Next i
    'Remove every other entry
    For i = 1 To 50
        TList1.Grid.RemoveRow i
    Next i
    'Clear all items.
End Sub
```

### See Also

**ColDelimiter** property

---

## AfterEditing Event

### Applies to

**TList** object

### Description

The **AfterEditing** event is triggered when the **ItemEditText** property is set to TLEDITMODE\_END or when editing is canceled by the user by clicking on another control, scrolling the control, etc.

### Syntax

```
Sub TList1_AfterEditing(ByVal ItemIndex As Long, ByVal vEditedText As Variant, vConvertedText As Variant, ByVal CancelledBy As Integer)
```

## Parameters

Parameter	Description
ItemIndex	Index of the item being edited.
vEditedText	Contains the string representation of data as just entered by the end-user.
vConvertedText	Contains the edited text converted by TList accordingly to the cell data type. This data will be placed into the corresponding grid cell. This may differ from <i>EditedText</i> due to Formatting.
CancelledBy	Set to 0 if editing was canceled by setting of one of the properties, set to 1 if editing was canceled by user.

**Note:** While editing checkboxes AfterEditing event will be fired with *vEditedText* parameter that contains a checkbox value but a checkbox visible caption. This value also can be addressed by **ItemCheckBoxValue** property and can be set to 0, 1 or 2 for standard checkboxes. For using non-standard values for checkboxes see **UncheckedValue**, **CheckedValue** and **GrayedValue** properties of **TListCheckbox** object.

## Example

The following sample code illustrates how to discard any editing changes:

```
Private Sub TList1_AfterEditing(ByVal ItemIndex As Long, ByVal EditedText As Variant, vConvertedText As Variant,
ByVal CancelledBy As Integer)

    Dim ShouldDiscard As Boolean
    ...
    ' place some code here to decide if you wish
    ' to discard changes - set ShouldDiscard
    ...
    ' return to original
    If ShouldDiscard Then
        vConvertedText = TList1.List(ItemIndex)
    EndIf

End Sub
```

It is possible to force resumption of editing within the **AfterEditing** Event (for example, after a wrong value is entered by an end user), by setting the **ItemEditText** property to TLEDITMODE\_CONTINUE ( = 3)

## Example

```
Private Sub TList1_AfterEditing(ByVal ItemIndex As Long, ByVal EditedText As Variant, vConvertedText As Variant,
ByVal CancelledBy As Integer)

    'new text can't be shorter than 5 characters
    If Len(vEditedText) < 5 Then
        MsgBox "The text size should be at least 5 characters"
        TList1.ItemEditText( -1 ) = TLEDITMODE_CONTINUE
        'continue editing of this item
    End If

End Sub
```

(Note that the index parameter of the ItemEditText property is ignored when setting this property to TLEDITMODE\_CONTINUE

## Remarks

Setting the **ItemEditText** to TLEDITMODE\_CANCEL does not trigger this event.

---

## Align Property

### Applies to

TList object

### Description

Returns or sets a value that determines whether a list box can appear in any size anywhere on a form, or whether it appears at the top or bottom of the form automatically sized to fit the form's width.

### Syntax

```
[form.]TList1.Align [= enum%]
```

### Settings

The **Align** property settings are:

Setting	Description
0	(Default) None - size and location can be set at design time or in code.
1	Top - list box is at the top of the form and its width is equal to the form's ScaleWidth property setting.
2	Bottom - list box is at the bottom of the form and its width is equal to the form's ScaleWidth property setting.

For more information, see the description of the **Align** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Integer

---

## AllowResizing Property

### Applies To

TListGrid object

### Description

Returns or sets a value that determines whether the user should be allowed to resize rows and columns in the TList control.

### Syntax

```
TListGridObject.AllowResizing [= enum% ]  
TList.Grid. AllowResizing [= enum% ]  
TList.ItemGrid(ItemIndex). AllowResizing [= enum% ]
```

### Settings

The **AllowResizing** property settings are:

Constants	Setting	Description
	<b>g</b>	
TLRESIZING_NONE	0	None. The user can't resize.
TLRESIZING_COLS	1	The user can resize columns.
TLRESIZING_TREEGRIDCAPTION	2	The user can change the height of TreeGrid caption.. Note: not applicable for ItemGrid objects.
TLRESIZING_COLS_AND_TREEGRIDCAPTION	3	The user can resize columns and change the height of TreeGrid caption using the mouse.
TLRESIZING_ROWS	4	The user can resize all rows except the grid caption one.

TLRESIZING_COLS_AND_ROWS_AND_TREEGRIDCAPTION	7	The user can resize all rows, all columns and change the height of TreeGrid caption.
TLRESIZING_ITEMGRIDCAPTION	8	The user can change the height of ItemGrid caption.
TLRESIZING_COLS_AND_ITEMGRIDCAPTION	9	The user can resize all columns and change the height of ItemGrid caption.
TLRESIZING_ROWS_AND_ITEMGRIDCAPTION	12	The user can resize all rows and change the height of ItemGrid caption.
TLRESIZING_COLS_AND_ROWS_AND_ITEMGRIDCAPTION	13	The user can resize all rows, all columns and change the height of ItemGrid caption.

### Remarks

To resize rows or columns, the mouse must be over the fixed area of the TList control, and close to a border between rows and columns. The mouse pointer will change into an appropriate sizing pointer and the user can drag the row or column to change the row height or column width.

### Example

The following code enables resizing for both rows and columns:

```
Sub Form1_Load ()
    TList1.Grid.AllowResizing = TLRESIZING_COLS_AND_ROWS_AND_TREEGRIDCAPTION
End Sub
```

### Data Type

Integer

---

## Appearance Property

### Applies to

TList object

### Description

Returns or sets the paint style of TList on an MDIForm or Form object at run time. Read-only at run time.

### Syntax

```
TList1.Appearance
```

### Settings

The **Appearance** property settings are:

Setting	Description
0	Flat. Paints TList without visual effects.
1	(Default) 3D. Paints TList with three-dimensional effects.

### Remarks

If set to 1 at design time, the **Appearance** property draws controls with three-dimensional effects.

### Data Type

Integer

---

## Appearance Properties (TListCheckbox Object)

### Applies to

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## TListCellDef object

### Description

The checkbox **Appearance** property determines whether TList will show default images for checkboxes in 2-d or 3-d presentation, or whether a custom image is being show (as set by the CheckedPicture, Unchecked Picture, or GrayedPicture properties)

### Values

The **Appearance** property settings are:

Constant	Description
TLCHK_APPEAR_2D	TList uses standard 2-D checkbox images
TLCHK_APPEAR_3D	TList uses standard 3-D checkbox images
TLCHK_APPEAR_CUSTOM	TList uses custom pictures as specified by CheckedPicture, UncheckedPicture or GrayedPicture properties.

### Syntax

```
[TListCellDef].EditInfo.CheckBox.Appearance [= Variant]
```

### Data Type

Integer

### Notes

TList will automatically reset the checkbox Appearance property to TLCHK\_CUSTOM\_PICTURE after any of the CheckedPicture, UncheckedPicture or GrayedPicture properties are set. To Reset all checkbox pictures for the cell(s) set the checkbox Appearance property to either 2-d or 3-d.

---

## AutoDragComplete Event

### Applies to

TList object

### Description

This event is generated just before the drag/drop process ends (in AutoDragDrop mode) and allows control over drag/drop completion, destination and data transfer.

### Usage

```
Sub TList_AutoDragComplete( ByVal SourceControl As TList, ByVal SourceItem As Long, ByVal TargetItem As Long, Operation As Integer, Cancel As Integer)
```

### Remarks

The **AutoDragComplete** event syntax has these parts:

Part	Description
Index	An integer that uniquely identifies a control if it is in a control array.
SourceControl	The TList control that initiated the drag/drop operation.
SourceItem	The index of an item being dragged during the operation. There is a list of predefined values, which you can use as an index (in multiple selection

	mode only): -1     drag all items of this tree -2     drag no items (but drag/drop operation will take place anyway) -3     drag all selected items -4     drag selected item
TargetItem	The index of an item that will be used as the destination for the drag/drop.
Operation	The type of a drag/drop operation that will be performed. Valid settings include : 0 – copy items from the source to the destination 1 – move items from the source to the destination
Cancel	This variable should be set to True to cancel the drag/drop operation.

### Sample

The following example illustrates how to disable moving root items within the same control. The copy operation is allowed in this case:

```
Private Sub TList1_AutoDragComplete(ByVal SourceControl As TList, ByVal SourceItem As Long,
    ByVal TargetItem As Long, Operation As Integer, Cancel As Integer)

    ' Check if it is a move operation
    If Operation = 1 Then
        If SourceControl = TList1 Then
            Cancel = True
        EndIf
    EndIf

End Sub
```

---

## AutoDragRequest Event

### Applies to

TList object

### Description

This event is generated just before the drag/drop process starts in AutoDragDrop mode and provides control over initiation of the drag/drop operation.

### Syntax

```
Sub TList_AutoDragRequest([Index As Integer,], SourceItem As Long, Cancel As Integer)
```

### Remarks

The **AutoDragRequest** event syntax has these parts:

Part	Description
Index	An integer that uniquely identifies a control if it is in a control array.
SourceItem	The index of an item being dragged during the operation. Note you can set this parameter to any valid item index. There is a list of predefined values, which you can use as an index (in multiple selection

	mode only): -1      drag all items of this tree -2      drag no items (but drag/drop operation will take place anyway) -3      drag all selected items -4      drag selected item
Cancel	This variable should be set to True to cancel the drag/drop operation.

### Sample

The following example illustrates how to disable drag/drop operations for all root items:

```
Private Sub TList1_AutoDragRequest(SourceItem As Long, Cancel As Integer)

    If SourceItem >= 0 Then
        Cancel = If(TList1.Shift(SourceItem) = 0, True, False)
    EndIf
    ...
End Sub
```

---

## AutoDragMode Property

### Applies to

TList object

### Description

This property determines how source items will be dropped (as child or as sibling) when using the automatic drag/drop mechanisms.

### Syntax

```
[form.]TList.AutoDragMode [= enum%]
```

### Settings

The **AutoDragMode** property settings are:

Setting	Description
0	(Default) None.
1	Drop as child
2	Drop as sibling (before item)
3	Drop as sibling (after item)

### Remarks

Setting AutoDragMode property to 0 doesn't cancel the automatic drag/drop operations. To cancel the Automatic Drag/Drop once it has begun, set the Cancel parameter in either the AutoDragRequest or the AutoDragComplete event, or set the standard **Drag** to 0 outside of these events.Data Type.

Please note that the **AutoDragMode** property takes effect on behavior of the **SmartDragDrop** property.

Long

---

## AutoExpand Property

### Applies to



**TList** object

### Description

The setting of the **AutoExpand** property determines how TList reacts to a click on the plus minus pictures or a double click on an item's text.

### Syntax

```
[form.]TList.AutoExpand [= enum%]
```

### Settings

The **AutoExpand** property settings are:

Setting	Description
0	None
1	(Default) Click on plus/minus or double click on item expands/collapses TList item.
2	Click on plus/minus on item expands/collapses TList item.
3	Double click on item's expands/collapses TList item.

### Data Type

Integer

---

## AutoFillColTitles and AutoFillRowTitles Properties

### Applies To

**TListGrid** object

### Description

The setting of the **AutoFill...** properties determines what default text will be shown in Grid column titles (**AutoFillColTitles** property) and row titles (**AutoFillRowTitles** property).

---

The AutoFillRowTitles now supports Automatic Hierarchic Numbering.

---

### Syntax

```
[form.]TList.Grid.AutoFillColTitles [= enum%]  
[form.]TList.Grid.AutoFillRowTitles [= enum%]
```

### Settings

The **AutoFillColTitles** and **AutoFillRowTitles** properties settings are:

Name	Setting	Description
TL_FILL_ROW_TITLES_NONE TL_FILL_COL_TITLES_NONE	0	None.
TL_FILL_ROW_TITLES_NUMBERS TL_FILL_COL_TITLES_NUMBERS	1	Numbers. (Default for <b>AutoFillRowTitles</b> ). TList will automatically number rows or columns starting with 1, 2, 3. .
TL_FILL_ROW_TITLES_CHARACTERS TL_FILL_COL_TITLES_CHARACTERS	2	Characters. (Default for <b>AutoFillColTitles</b> ). TList will automatically label rows or columns alphabetically(A, B, C ... AA, AB, etc.) (The English alphabet is used regardless of Windows language settings).
TL_FILL_ROW_TITLES_HIERARCHIC	3	Hierararchic. TList will automatically label rows in a Tree structure to reflect the hierarchic structure. 1, 1.1, 1.2, 1.2.1, 1.2.2, 1.3, ....

## Data Type

Integer

---

# AutoNewPage Property

## Applies To

TListReport object

## Description

Specifies if TList should automatically start a new page while printing if there is a need. Otherwise it is up to the programmer to specify a new page (for example, using Printer object's NewPage method).

Not available at design-time.

## Syntax

```
TListReportObject.AutoNewPage[= boolean% ]  
[form.]TList.Report.AutoNewPage [= boolean% ]
```

## Remarks

If the **PrinterObject** property is set with the DC of a physical printer, and the **AutoNewPage** property is set to True, TList calls the **NewPage** method automatically for a given output DC after completion of the **EndPage** event. If the **AutoNewPage** property is set to False you should take care of initializing a new page manually (programmatically).

## Data Type

Integer (boolean)

---

# AutoScrDuringDragDrop Property

## Applies to

TList object

## Description

The setting of **AutoScrDuringDragDrop** determines whether TList will automatically scroll the outline during Drag Drop operations.

## Syntax

```
[form.]TList.AutoScrDuringDragDrop[= enum%]
```

## Settings

The **AutoScrDuringDragDrop** property settings are:

Setting	Description
0	(Default) None - no scrolling occurs during drag-drop.
1	Horizontal - scrolling occurs only in horizontal direction.
2	Vertical - scrolling occurs only in vertical direction.
3	Both - scrolling occurs only in both directions.

## Data Type

Integer

---

# AutoSizeRow Method, AutoSizeColumn Method

## Applies to

TListGrid object

## Description

The AutoSize methods provide a mechanism to resize rows or columns to fit the data held within a TList grid.

- The **AutoSizeRow** method automatically resets the height of a specified row to fit the maximum height of the data in the row, and returns the new row height in pixels.
- The **AutoSizeColumn** method automatically resets the width of a specified column to fit the maximum width of the data in the column and returns the new column width in pixels.

## Syntax

```
INewHeight = TListGrid.AutoSizeRow ( IRowIndex )  
INewWidth = TListGrid.AutoSizeColumn ( IColumnIndex, AutoSizeFlags )
```

## Parameters

Name	Description
LRowIndex	Specifies the row number within a grid
IColumnIndex	Specifies a column number within a grid.
AutoSizeFlags	Specifies what data should be considered when calculating the width.

AutoSizeFlags parameter's flags:

Constant	Value	Description
GRID_AUTOSIZECOL_DEFAULT	&H0	Set column size to widest visible cell in the column. Note - Width is limited to width of TList window.
GRID_AUTOSIZECOL_INCLUDE_INVISIBLE_CELLS	&H08	Width of invisible cells ( cells held by collapsed and or hidden items) in the column are taken into the account when autosizing column  * * * This is ignored if specifying GRID_AUTOSIZECOL_ONLY_CURRENTLY_VISIBLE_CELLS flag.
GRID_AUTOSIZECOL_EXCLUDE_TITLE_CELL	&H10	Ignore width of column title cell when autosizing column.
GRID_AUTOSIZECOL_ONLY_CURRENTLY_VISIBLE_CELLS	&H20	Only CURRENTLY visible in TList window items take into account. Items scrolled out of view are ignored from autosizing calculations.  This is generally desirable when working with VirtualItems.
GRID_AUTOSIZECOL_UNLIMITED_WIDTH	&H40	Width of the column is not limited by width of TList window.

## Remarks

- The AutoSizing mechanism in TList can NOT support RTF formatted data.
- The AutoSizeOptions property may also be used to instruct TList to automatically resize columns when the user double clicks on a Grid Column Title.

## Example

```
NewHeight = TList1.Grid.AutoSizeRow ( 134 )  
NewWidth = TList1.Grid.AutoSizeColumn ( 122 , 0 )
```

## See Also

**AutoSizeOptions** Property

---

# AutoSizeOptions Property

## Applies to

**TListGrid** object

## Description

The **AutoSizeOptions** property specifies whether TList will automatically resize grid columns to fit the contained data in response to a double click on the separator in the column title cells between columns.

## Syntax

```
TListGrid.AutoSizeOptions = Desired Setting
```

## Values

The property may be set to 0, (cancel any auto-sizing) or to a Logical OR combination of any of the following settings:

Constant	Value	Description
GRID_AUTOSIZE_COLUMN_ON_SEPARATOR_DOUBLECLICK (set by default)	&H01	Set column width depending on the size of the widest visible cell in that column. Note: width of the column is limited to the width of TList window.
GRID_AUTOSIZE_INCLUDE_INVISIBLE_CELLS	&H08	Width of invisible cells ( cells held by collapsed and or hidden items) in the column are taken into the account when autosizing column Note: this flag is ignored if the following flag is set: GRID_AUTOSIZECOL_ONLY_CURRENTLY_VISIBLE_CELLS.
GRID_AUTOSIZE_EXCLUDE_TITLE_CELL	&H10	Ignore the width of the column title cell when calculate automatically the width of the column.
GRID_AUTOSIZE_ONLY_CURRENTLY_VISIBLE_CELLS	&H20	Only CURRENTLY visible in TList window items take into account. Items scrolled out of view are ignored from autosizing calculations. Note: this flag is usually helpful when working with virtual items.
GRID_AUTOSIZE_UNLIMITED_WIDTH	&H40	The width of the column will not be limited by the width of TList

		window.
GRID_AUTOSIZE_ROW_ON_SEPARATOR_DOUBLECLICK (set by default)		When this flag is set and TLRESIZING_ROWS flag of Grid.AllowResizing property is set as well, TList automatically resizes grid rows in order to fit the content in response to a double click on the separator between row title cells.

### Remarks

- TList will adjust the column width as required in either direction – either reducing or enlarging column width to fit the data.
- TList associates each column with the Column separator to it's right. To automatically resize the column width double click on the column separator to the right
- The AutoSizing mechanism in TList can not currently support RTF formatted data.

### Example

```
' Turn on Automatic Resizing Feature, include cells which may be currently collapsed
TList1.Grid.AutoSizeOptions =GRID_AUTOSIZE_COLUMN_ON_SEPARATOR_DOUBLECLICK _
Or GRID_AUTOSIZE_INCLUDE_INVISIBLE_CELLS

' Turn Off Automatic Resizing Feature
TList1.Grid.AutoSizeOptions 0
```

### See Also

AutoSizeColumn and AutoSizeRow methods

---

## BackColor and DefItemCellBackColor Property

### Applies To

TList control

TListCellDef object

### Description

TLists's **BackColor** property determines the default color displayed in background of an item.

TListCellDef's property determines the default color displayed in background of a cell.

DefItemCellBackColor determines the default color displayed in the background of an item cell.

### Syntax

```
[form.]TList.DefItemCellBackColor[ = color&]
[form.]TList.BackColor[ = color&]
[form.]TList.ItemCell(ItemIndex&).CellDef.BackColor[ = color&]
[form.]TList.Grid.GridCellDef. BackColor[ = color&]
[form.]TList.Grid.Cells(Row&, Col&). BackColor[ = color&]
[form.]TList.Grid.RowTitleCellDef. BackColor[ = color&]
[form.]TList.Grid.ColTitleCellDef. BackColor[ = color&]
```

### Remarks

Setting of this property updates the control display unless the **Redraw** property is set to False.

If applied to a **LevelDef** object Item, the property refers to the default Background Color of cells at the specified hierarchic level (see *Visual Elements* chapter).

### Example

```
TList1.DefItemCellBackColor = RGB(127, 0, 127)
TList1.DefItemCellBackColor = QBColor(2)
TList1.BackColor = RGB(127, 0, 127)
TList1.BackColor = QBColor(2)
```

### Data Type

Long

---

## BackColor and SelBackColor Properties (TListNode Object)

### Applies to

TListNode object

### Description

The **BackColor** property determines the background color of the item associated with this **TListNode Object**. The **SelBackColor** property determines the selected background color of the item associated with this **TListNode Object**.

### Syntax

```
Node.BackColor[ = color&]  
Node.SelBackColor[ = color&]
```

### Data Type

Long

### See Also

TListNode object properties

---

## BackColorBkg Property

### Applies To

TListGrid object

### Description

**BackColorBkg** property determines the background color of the control area not occupied with Tree Grid. This property works for TreeGrid objects only.

### Syntax

```
[form.]TList.Grid.BackColorBkg[ = color&]
```

### Remarks

Setting of this property updates the control display unless the **Redraw** property is set to False.

### Example

```
TList1.BackColorBkg = RGB(127, 0, 127)
```

### Data Type

Long

---

## BackPicture and BackPictureAlignment Properties

### Applies to

TList object

### Description

The **BackPicture** property determines what graphic will be displayed on TList's background. The **BackPictureAlignment** property specifies how the picture will be displayed.

### Syntax

```
[form.]TList.BackPicture [ = Picture Object]  
[form.]TList.BackPictureAlignment [ = Enum%]
```

### Settings

The **BackPictureAlignment** property settings are:

Constant	Settings	Description
TLBACKPICTUREALIGNMENT_LEFT_TOP	0	Image is left/top aligned.
TLBACKPICTUREALIGNMENT_LEFT_MIDDLE	1	Image is left/middle aligned.
TLBACKPICTUREALIGNMENT_LEFT_BOTTOM	2	Image is left/bottom aligned.
TLBACKPICTUREALIGNMENT_RIGHT_TOP	3	Image is right/top aligned.
TLBACKPICTUREALIGNMENT_RIGHT_MIDDLE	4	Image is right/middle aligned.
TLBACKPICTUREALIGNMENT_RIGHT_BOTTOM	5	Image is right/bottom aligned.
TLBACKPICTUREALIGNMENT_CENTER_TOP	6	Image is center/top aligned.
TLBACKPICTUREALIGNMENT_CENTER_MIDDLE	7	Image is center/middle aligned.
TLBACKPICTUREALIGNMENT_CENTER_BOTTOM	8	Image is center/bottom aligned.
TLBACKPICTUREALIGNMENT_STRETCH	9	Image is stretched to fit image area.
TLBACKPICTUREALIGNMENT_TILE	10	Image is tiled in the image area.

---

## BackwardCompatible Property

### Applies to

TList object

### Description

This property is no longer supported. It had been added in TList 3 to temporarily provide backward compatibility with earlier versions. To simulate the effect, simply call the click event of TList within the **PictureClick** event.

---

## BeforeDrag Method

### Applies to

TList object

### Description

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

This property was added in TList to provide the Drag Drop functionality of earlier versions of TList. Call this method before initiating Dragging of TList elements. .

### Syntax

```
[form.]TList1.BeforeDrag
```

### Example

```
TList1.BeforeDrag  
TList1.Drag 1
```

---

## BeforeItemDeactivate, BeforeGridRowDeactivate and BeforeGridCellDeactivate Events

### Applies to

TList object

### Description

TList 8 now supports a set of events allowing interception of a change in activation of items/cells before the change occurs - for instance before deactivating one item/cell and moving to ( activating ) a new item in List or Tree presentation, or a new row or cell in Grid presentation.

The new events, BeforeItemDeactivate, BeforeGridRowDeactivate and BeforeGridCellDeactivate occur before the currently active item, row, or cell is deactivated by the user's mouse or keyboard action.

The purpose of these events is to allow the programmer to recognize that a change in activation is about to take place, to take any necessary actions before that change takes place, and to optionally prevent the change.

Each of these events has 4 parameters to determine

- the currently active item, row or grid cell
- the item, row, or grid cell about to be activated
- the input type – whether mouse or keyboard action is initiating the change
- a cancel parameter to allow the user's action to be cancelled / ignored

### Syntax

```
Sub TList1_BeforeItemDeactivate( _  
    ByVal iCurrentItemIndex As Long, _  
    ByVal iNewItemIndex As Long, _  
    ByVal iInputType As Integer, _  
    iCancelFlags As Long)  
  
Sub TList1_BeforeGridRowDeactivate( _  
    ByVal objCurrentGridRow As TListProLibCtl.TListRowDef, _  
    ByVal objNewGridRow As TListProLibCtl.TListRowDef, _  
    ByVal iInputType As Integer, _  
    iCancelFlags As Long)  
  
Sub TList1_BeforeGridCellDeactivate( _  
    ByVal objCurrentGridCell As TListProLibCtl.TListGridCell, _  
    ByVal objNewGridCell As TListProLibCtl.TListGridCell, _  
    ByVal iInputType As Integer, _  
    iCancelFlags As Long)
```

### Parameters

For **BeforeItemDeactivate** event the parameters are:

Parameter	Description
<b>CurrentItemIndex</b>	Returns the index of the currently active item



<b>NewItemIndex</b>	(-1 if no item is currently active). Returns the index of the item being activated/selected.
<b>InputType</b>	Returns a value specifying the type of end-user activity triggering the event.
<b>CancelFlags</b>	Determines whether the user's action should be cancelled, whether the selection and activation state should be changed by the user action.

For **BeforeGridRowDeactivate** event the parameters are:

Parameter	Description
<b>ObjCurrentGridRow</b>	Returns a reference to the grid row object that is currently active (Nothing if there is no initially active Grid Row).
<b>ObjNewGridRow</b>	Returns a reference to the grid row object that is being activated/selected.
<b>InputType</b>	Returns a value specifying the type of end-user activity triggering the event.
<b>CancelFlags</b>	Determines whether the user's action should be cancelled, whether the selection and activation state should be changed by the user action.

For **BeforeGridCellDeactivate** event the parameters are:

Parameter	Description
<b>ObjCurrentGridCell</b>	Returns a reference to the grid cell object that is currently active (Nothing if there is no initially active GridCell).
<b>objNewGridCell</b>	Returns a reference to the grid cell object that is being activated/selected.
<b>InputType</b>	Returns a value specifying the type of end-user activity triggering the event.
<b>CancelFlags</b>	Determines whether the user's action should be cancelled, whether the selection and activation state should be changed by the user action.

## Settings

For each event, the **InputType** parameter returns one of the following values:

Setting	Description
TL_BEFOREACTIVE_MOUSEINPUT = 0	This event was triggered by mouse action.
TL_BEFOREACTIVE_KEYBOARDINPUT = 1	This event was triggered by keyboard action.

For each event, the **CancelFlags** parameter may be set to a logical OR combination of the following flag values:

Setting	Description
TL_BEFOREDEACTIVE_CANCELMOUSESELECTION = 1	If this flag specified, no change in selection will occur in response to mouse input, Navigation to (Activation of) the new item, and triggering of the activation events may still occur depending on whether or not TL_BEFOREACTIVE_CANCELMOUSENAVIGATION flag is set, in the ICancelFlags parameter.
TL_BEFOREDEACTIVE_CANCELMOUSENAVIGATION = 2	If this flag specified, no navigation/activation will occur in response to mouse input and no corresponding events (Click, Deactivate/Activate) will be triggered.
TL_BEFOREDEACTIVE_CANCELKEYBOARDSELECTION = 4	If this flag specified, no change in selection will occur in response to keyboard input, Navigation to (Activation of) the new item, and triggering of the activation events may still occur depending on whether or not TL_BEFOREACTIVE_CANCELKEYBOARDNAVIGATION flag is set, in the CancelFlags flags parameter.
TL_BEFOREDEACTIVE_CANCELKEYBOARDNAVIGATION = 8	If this flag specified, Navigation will not occur in response to the keyboard input, no corresponding events (Click, Deactivate/Activate) will be triggered.
TL_BEFOREDEACTIVE_CANCELALL = 65535	If this flag specified neither Selection nor Navigation will occur in response to either mouse or keyboard actions.

### Remarks

Note that only one of the above events will be triggered in response to a user action (mouse click or keyboard action).

The event triggered depends on what type of item/cell the user is moving to.

- If the user is attempting to navigate to an ordinary item in List or Tree (not as part of a Grid) then the BeforeItemDeactivate event would be triggered.
- If the user is attempting to navigate to a row in a grid where Navigation Mode = Row, then the BeforeGridRowDeactivate event would be triggered.
- If the user is attempting to navigate to a cell in a grid where Navigation Mode = Cell, then the BeforeGridRowDeactivate event would be triggered

These events are not meant to be used in conjunction with ItemGrids and such use is not formally supported. This should work but is not guaranteed in all cases of use and may also be confusing. If you have items in the tree structure along with ItemGrids for example, a user moving from an ordinary tree item (not in a grid) to row or cell belonging to an ItemGrid would then trigger BeforeGridRowDeactivate or BeforeGridCellDeactivate. If he then clicks back on normal tree item (not in grid) then the BeforeItemDeactivate is triggered. Also if there are ItemGrid objects, it is possible that the originally active object and the one user are moving to may be different types of objects. This can make first parameter of the triggered event come up empty if going from an item to a row or cell.

When navigation is cancelled the normal MouseDown/MouseUp and KeyDown/KeyUp events may still be triggered, but ItemDeactivate, ItemActivate, Click, DblClick, ItemClick events will be preempted and will not be triggered.

### Example

```
Sub TList1_BeforeItemDeactivate( _
    ByVal ICurrentItemIndex As Long, _
    ByVal INewItemIndex As Long, _
    ByVal InputType As Integer, _
    ICancelFlags As Long)
```

```

' Identify Mouse or Keyboard Action
If InputType = 0 Then
    InputType = "Mouse"
Else
    InputType = "Keyboard"
End If

'Add listbox item to identify what user is trying to do
lstStatus.AddItem _
    " User trying to move " & _
    " from ItemIndex =" & ICurrentItemIndex & _
    " to ItemIndex =" & INewItemIndex & _
    " using the " & InputType

' If user has not filled in textbox don't allow him to move
If Text1.text = "" then
    ICancelFlags = TL_BEFOREDEACTIVE_CANCELALL = 65535
End If
End Sub

```

## BeginPage Event

### Applies to

TList object

### Description

This event is generated during the printing process just before TList starts to print a next page. The user can handle this event to print any additional information on a printer DC (for example page header).

### Usage

```
Sub TList_BeginPage([Index As Integer,] PrinterObject As Variant, ReportPage As TListReportPage)
```

### Remarks

The **BeginPage** event syntax has these parts:

Part	Description
Index	An integer that uniquely identifies a control if it is in a control array.
PictureObject	An output device, can be a printer object, a picture-box, or a DC handle
ReportPage	An object that describes the current page, see TListReportPage properties and methods

## BorderColor and DefItemCellBorderColor Properties

### Applies To

TList control

TListCellDef object

### Description

Determines the default border color displayed around an item cell.

### Syntax

```
[form.]TList.ItemCell(ItemIndex&).CellDef.BorderColor[ = color&] - sets border color for a specific cell.
[form.]TList.Grid.GridCellDef.BorderColor[ = color&] - sets default border color for cells in a grid
[form.]TList.Grid.Cells(Row&, Col&).BorderColor[ = color&] - sets border color for a specific cell in a grid.
[form.]TList.Grid.RowTitleCellDef.BorderColor[ = color&] - sets default border color for cells in the first column of a grid
[form.]TList.Grid.ColTitleCellDef.BorderColor[ = color&] - sets default border color for cells in the first row of a grid.
[form.]TList.DefItemCellBorderColor[ = color&] - sets default border color for cells in a row not in a grid.
```

### Remarks

Setting of these properties updates the control display unless the **Redraw** property is set to False.

### Example

```
TList1.DefItemCellBorderColor = RGB(127, 0, 127)
TList1.DefItemCellBorderColor = QBColor(2)
```

### Data Type

Long

## BorderStyle and DefItemCellBorderStyle Properties

### Applies To

TList control

TListCellDef object

### Description

TLists's **BorderStyle** property determines the border style of a TList box.

TListCellDef's **BorderStyle** property determines what borders will be drawn around an Item cell.

DefItemCellBorderStyle determines the default border style which will be used for an Item cell.

### Syntax

```
[form.]TList.BorderStyle [ = enum%]
[form.]TList.DefItemBorderStyle = [enum%]
[form.]TList.ItemCell(ItemIndex&).CellDef.BorderStyle [ = enum%]
[form.]TList.Grid.GridCellDef. BorderStyle [ = enum%]
[form.]TList.Grid.Cells(Row&, Col&).BorderStyle [ = enum%]
[form.]TList.Grid.RowTitleCellDef. BorderStyle [ = enum%]
[form.]TList.Grid.ColTitleCellDef. BorderStyle [ = enum%]
```

### Remarks

The TList's **BorderStyle** property settings are:

Setting	Description
0	No border
1	Fixed single border

For more information, see the description of the **BorderStyle** property in *Microsoft Visual Basic On-Line Help*.

The **DefItemCellBorderStyle** and **CellDef.BorderStyle** properties settings are:

Constant	Setting	Description
TLBORD_NONE	0	(Default) No border.
TLBORD_SINGLE	1	Single one-pixel border.
TLBORD_DOUBLE	2	2-pixel border.
TLBORD_INSET	3	3-D inset border.
TLBORD_OUTSET	4	3-D outset border.

### Data Type

Integer

# BorderStyle Property (TListGrid Object)

## Applies to

TListGrid object

## Description

This property allows the developer to change the border style of grid objects. So it is possible to specify the visibility of column titles border, row titles border and outer border of the grid separately.

## Syntax

```
Grid.BorderStyle = Flags&
```

## Data Type

Long

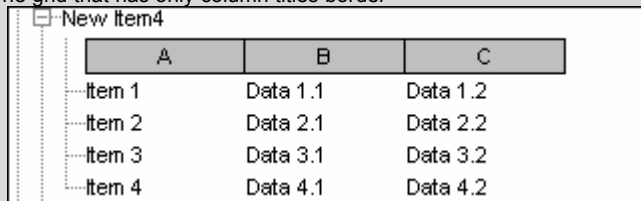
## Settings

Constant	Value	Description
TL_GRIDBRD_BORDER	&H1	Draws a border only around the outer bounds of the grid object.
TL_GRIDBRD_COLTITLES	&H2	Draws a border only around the column titles of the grid object (column separators are drawn also).
TL_GRIDBRD_ROWTTITLES	&H4	Draws a border only around the row titles of the grid object (row separators are drawn also).

## Example

Here are several screenshots showing typical grid views:

The grid that has only column titles border

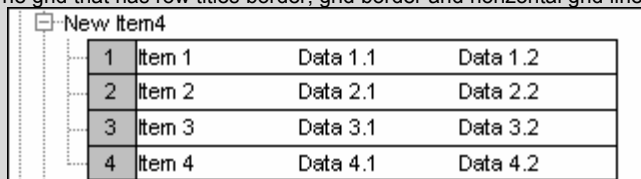


	A	B	C
Item 1	Data 1.1	Data 1.2	
Item 2	Data 2.1	Data 2.2	
Item 3	Data 3.1	Data 3.2	
Item 4	Data 4.1	Data 4.2	

corresponding VB code:

```
ItemGrid(1).BorderStyle = TL_GRIDBRD_COLTITLES  
ItemGrid(1).GridLinesStyle = TLGRIDLINES_NONE
```

The grid that has row titles border, grid border and horizontal grid lines

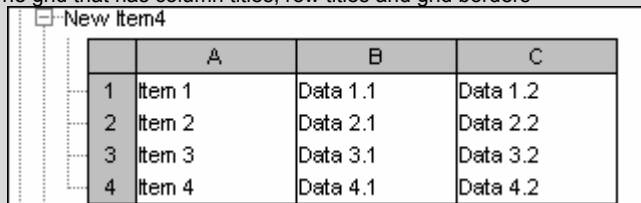


1	Item 1	Data 1.1	Data 1.2
2	Item 2	Data 2.1	Data 2.2
3	Item 3	Data 3.1	Data 3.2
4	Item 4	Data 4.1	Data 4.2

corresponding VB code:

```
ItemGrid(1).BorderStyle = TL_GRIDBRD_ROWTTITLES Or TL_GRIDBRD_BORDER  
ItemGrid(1).GridLinesStyle = TLGRIDLINES_HORIZONTAL
```


The grid that has column titles, row titles and grid borders

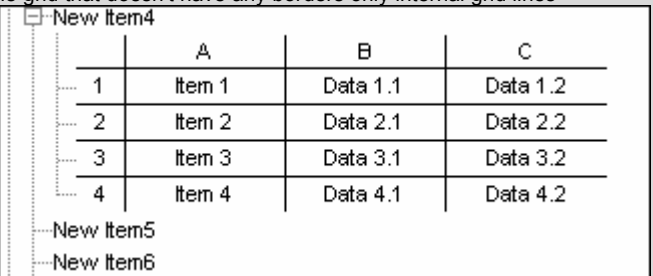


	A	B	C
1	Item 1	Data 1.1	Data 1.2
2	Item 2	Data 2.1	Data 2.2
3	Item 3	Data 3.1	Data 3.2
4	Item 4	Data 4.1	Data 4.2

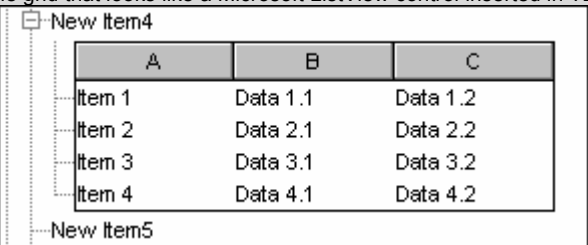
corresponding VB code:

```
ItemGrid(1).BorderStyle = TL_GRIDBRD_COLTITLES Or TL_GRIDBRD_ROWTTITLES Or
```

 `TL_GRIDBRD_BORDER`  
`ItemGrid(1).GridLinesStyle = TLGRIDLINES_NONE`  
 The grid that doesn't have any borders only internal grid lines



corresponding VB code:  
`ItemGrid(1).BorderStyle = 0`  
`ItemGrid(1).GridLinesStyle = TLGRIDLINES_INTERNAL_BOTH`  
 The grid that looks like a Microsoft ListView control inserted in TList



corresponding VB code:  
`ItemGrid(1).BorderStyle = TL_GRIDBRD_BORDER Or TL_GRIDBRD_COLTITLES`  
`ItemGrid(1).GridLines = TLGRIDLINES_NONE`

## BottomIndex Property

### Applies to

TList object

### Description

The **BottomIndex** property returns the index of the last item displayed within the control window. Read only at run-time, not available at design time.

### Syntax

[form.]TList1.BottomIndex

### Data Type

Long

## Caption Property

### Applies to

TList object

### Description

TList's caption is shown at the top of the control. The visibility and form of the caption is controlled by the **ShowCaption** property.

### Syntax

[form.]TList1.Caption[=str\$]

### Data Type

String

---

## CellDef Property

### Applies To

TListLevelDef object  
TListGridCell object  
TListColDef object  
TListNode object

### Description

Returns a reference to a **TListCellDef** object. This object is used to collect all properties which determine how the object will be displayed.

**TListGridCell's CellDef** property is used to specify how data will be displayed in this grid cell.

**TListColCell's CellDef** property is used to specify default settings for all cells of a given column.

**TListLevelDef's CellDef** property is used to specify default settings for all items of a given indentation level.

### Syntax

```
[form.]TList1.LevelDefs(IndentationLevel&).CellDef  
[form.]TList1.Grid.ColDefs(ColDef&).CellDef  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs(ColDef&).CellDef  
[form.]TList1.ItemValues(ItemIndex&, ValueName$).CellDef  
[form.]TList1.Grid.Cells(Row&, Col&).CellDef
```

### Remarks

To discard all attributes which might be specified in a TListCellDef object, set the **CellDef** property to **Nothing**:

```
Set TList1.Grid.ColDefs(3).CellDef = Nothing
```

### Example

```
TList1.Grid.ColDefs(3).CellDef.BackColor = RGB(127, 127, 127)  
TList1.Grid.Cells(3, 2).CellDef.Font.Size = 9.3
```

---

## CellEdit Property

### Applies to

TList object  
TListGrid object

### Description

Setting the **CellEdit** property enables, concludes or cancels an in-place editing operation for some specified cell of the grid.

Upon setting this property to TLCELLEDIT\_BEGIN, the **GridCellRequestEditing** event will be triggered.

Upon setting this property to TLCELLEDIT\_END or TLCELLEDIT\_CANCEL, the **GridCellAfterEditing** event will be triggered.

Not available at design-time, write only at run-time.

### Syntax

```
TListGridObject.CellEdit(Row&, Col&) = %enum  
[form.]TList.Grid.CellEdit(Row&, Col&) = %enum  
[form.]TList.ItemGrid(index&).CellEdit(Row&, Col&) = %enum
```

### Data Type

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

Integer

### Settings

Constant	Value	Description
TLEDITMODE_CANCEL	0	Cancel editing of the grid cell.
TLEDITMODE_END	1	Conclude editing of the grid cell.
TLEDITMODE_BEGIN	2	Start editing of the grid cell.
TLEDITMODE_CONTINUE	3	Continue editing This setting may be used within the GridCellAfterEditing event.

### Note

The row and column parameter of the **CellEdit(Row,Column)** property are used only to initiate editing (when setting the property to 2). Otherwise (when canceling, completing, or resuming editing) the **Row** and **Column** parameters are ignored

### See Also

GridCellRequestEditing, GridCellAfterEditing, GridCellEditingKeyDown, GridCellEditingKeyUp and GridCellEditingKeyPress events, EditingMode property

---

## Cells Property

### Applies To

TListGrid object

### Description

Returns TListGridCell object for the specified cell.

---

TList 8 supports referencing TListColDefs and TListGridCells by ValueName as well as by column index.

---

### Syntax

```
Grid.Cells(Row&, Col&) [=TListGridCell]  
TListGridObject.Cells(Row&, Col&) [=TListGridCell]  
TList.Grid.Cells(Row&, Col&) [=TListGridCell]  
TList.ItemGrid(ItemIndex).Cells(Row&, Col&) [=TListGridCell]
```

### Remarks

To set a title for 5<sup>th</sup> column, use the following code:

```
TList1.Grid.Cells(0, 5).Value = "5th Column Title"  
' or using column value name  
TList1.Grid.Cells(0, "ColumnValueName").Value = "5th Column Title"
```

To set a title for 5<sup>th</sup> row, use the following code:

```
TList1.Grid.Cells(5, 0).Value = "5th Row Title"
```

---

## CellValue Property (TListComboItem Object)

### Applies to

TListComboItem object

### Description

The **CellValue** property of each **TListComboItem** object is a unique key identifying the **ComboBox** list item.



For In-Place Editing, The CellValue is used -

- as the value assigned to a TList cell when the associated combobox list item is selected
- as the default value to be displayed in the combobox drop down list for each item ( this display may be replaced by the value of the DisplayValue property if set )
- as the default text to be displayed in the text editing area of the combobox editing object as the user selects an item from the drop down portion – the CellValue of the combobox list item is shown unless a DisplayValue has been set
- to determine which combobox list item is initially shown as selected when combobox editing is initiated. The value of the cell before editing is compared with the CellValue properties of each combobox list item. The Combobox Item which matches the cell value is automatically selected when editing begins.

For Intelligent Formatting, TList compares the value of the cell with each the CellValue property of each combobox list item. When a match is found the cell is displayed using the combobox list item's colors, fonts, picture, even text as specified in the DisplayValue property of the combobox list item.

The value of this property is identical to the first parameter in the **Add** method of the **TListComboItems** object. It may also be set or read directly.

### Syntax

```
[TListCombobox].Items(Index).CellValue = [value]
```

### Example

```
'Specify use of a combobox for editing column 5 of a TList grid
TList1.Grid.ColDefs(5).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCLICK
TList1.Grid.ColDefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
With TList1.Grid.ColDefs(5).CellDef.EditInfo.ComboBox.Items
  'the 1st parameter of the Combobox item's ADD method sets the CellValue
  .Add "Very Urgent"
  .Add "Priority"
  .Add "Normal"
End With
' The user may now select an item from the combobox. Selecting the second item sets the cell's value to "Priority"
```

---

## CheckBox Property (TListEditInfo Object)

### Applies to

TListCellDef object

### Description

This property returns a **TListCheckBox** object controlling checkbox editing for a data cell or cells. Note: the **Style** property of **EditInfo** object must be set to TLEDITINFO\_CHECKBOX before using the Checkbox.

### Syntax

```
[TListCellDef].EditInfo.CheckBox
```

### Data Type

TlistCheckBox

---

# CheckboxValue Property

## Applies to

TListGridCel object

## Description

Returns or sets a value that determines the state of a checkbox in a grid cell.

## Syntax

```
TListGridCelObject.CheckboxValue [= Variant]  
[form.]TList.Grid.Cells(Row&, Col&).CheckboxValue [= Variant]  
[form.]TList.ItemGrid(ItemIndex).Cells(Row&, Col&).CheckboxValue [= Variant]
```

## Data Type

Variant

## Remarks

**CheckboxValue** determines the item checkbox state according to **CheckedValue/UncheckedValue/GrayedValue** properties. **CheckboxValue** is automatically updated in response to end-user actions( mouse-click / or doubleclick) on the item if **Editable** property is switched on.

The checkbox state for items in a simple tree or grid, or for cells in the first column of a grid, is equivalently read or set using the ItemCheckBoxValue property.

## Example:

```
TList1.Redraw = False'switch off for fast work  
'build tree  
  
TList1.AddItem "Root"  
TList1.AddItem "Item1", 0  
TList1.AddItem "Item2", 0  
TList1.AddItem "Item3", 0  
TList1.ItemGrid(0).Cols = 5'create grid  
  
'work with third grid column:  
With TList1.ItemGrid(0).ColDefs(3).CellDef.EditInfo  
  .Style = TLEDITINFO_CHECKBOX'Show checkboxes  
  .Editable = TLEDITABLE_ALWAYS_ONCLICK'edit on click  
  .CheckBox.CheckedValue = True""Yes" string as checked value  
  .CheckBox.UncheckedValue = False""No" string as unchecked value  
End With  
  
'By default all cells are unchecked  
'Next line switches checkbox to checked state in second row, third column:  
TList1.ItemGrid(0).Cells(2, 3).CheckboxValue = True  
TList1.Redraw = True 'switch on to show changes
```

## See Also:

ItemCheckBoxValue, EditInfo, Editable properties TListCheckBox object

---

# CheckedPicture, UncheckedPicture and GrayedPicture Properties (TListCheckbox Object)

## Applies to

TListCelDef object

TListCheckBox object

## Description

By default **TList** uses standard Windows images for checked, unchecked and grayed states of checkboxes.

The **CheckedPicture**, **UncheckedPicture** and **GrayedPicture** set alternative pictures for each checkbox state.

### Syntax

```
[TListCellDef].EditInfo.CheckBox.CheckedPicture [= Picture object]
[TListCellDef].EditInfo.CheckBox.UncheckedPicture [= Picture object]
[TListCellDef].EditInfo.CheckBox.GrayedPicture [= Picture object]
```

### Data Type

StdPicture

### Example

```
TListCellDef.EditInfo.CheckBox.CheckedPicture = _
    LoadPicture( "C:\some path\ok.ico" )
TListCellDef.EditInfo.CheckBox.CheckedPicture = Picture1.Picture
TListCellDef.EditInfo.CheckBox.CheckedPicture = TList1.MarkPicture(1)
```

### Remarks

Note that setting any of the **CheckedPicture**, **UncheckedPicture** or **GrayedPicture** properties automatically changes the value of the **Appearance** property to 3 (TLCHK\_APPEAR\_CUSTOM). The checkbox pictures may be reset to the standard checkbox images by setting the appearance property to either TLCHK\_APPEAR\_CUSTOM\_2D or TLCHK\_APPEAR\_CUSTOM\_3D.

The placement (above, below, left or right of text) of the checkbox or customized pictures may be controlled using the **Alignment** property of the **TListCellDef** object, or the **DefCellPictureAlignment** of the **TList** control itself. The size of the checkbox image (if not the default image) may also be controlled using the celldef **PictureWidth** and **PictureHeight** properties.

```
TList.Grid.Coldefs(4).CellDef.EditInfo.Style = TLEDITINFO_CHECKBOX
With TList.Grid.Coldefs(4).CellDef.EditInfo.CheckBox
    .CheckedPicture = loadpicture ( "c:\some path\checked.ico")
    .UncheckedPicture = loadpicture ( "c:\some path\unchecked.ico")
End With
'make all checkboxes in column 4 left aligned
TList.Grid.Coldefs(4).CellDef.PictureAlignment = TLPICTUREALIGNMENT_LEFT_CENTER
```

---

## CheckedValue, UncheckedValue and GrayedValue Properties (TListCheckbox Object)

### Applies to

**TListCellDef** object

**TListCheckBox** object

### Description

The **CheckedValue**, **UncheckedValue** or **GrayedValue** properties determine the **Value** of the cell in which a checkbox appears according to the state of the checkbox. Values

The Default setting for these properties are:

[TListCheckBox].**UncheckedValue** = 0

[TListCheckBox].**CheckedValue** = 1

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

[TListCheckBox].**GrayedValue** = 2

### Syntax

```
[TListCellDef].EditInfo.CheckBox.CheckedValue [= Variant]  
[TListCellDef].EditInfo.CheckBox.UncheckedValue [= Variant]  
[TListCellDef].EditInfo.CheckBox.GrayedValue [= Variant]
```

### Data Type

Variant

### Remarks

In response to checking or unchecking a checkbox, the **CheckBoxValue** property of the cell ( **ItemCheckBoxValue** property for items in a tree) is automatically set to either the Checked, Unchecked or Grayed value.

The checkbox state can be read by reading the **CheckBoxValue** property of the corresponding cell( **ItemCheckBoxValue** property for the items in the tree)

```
Dim chkValue as Long  
chkValue = TList.Grid.Cells(2,3).CheckBoxValue 'for grid cell  
chkValue = TList.ItemCheckBoxValue(15) 'for tree items  
chkValue = TList.Node(15).CheckBoxValue 'also for tree items
```

The checkbox state can be set by setting the **CheckBoxValue** (or **ItemCheckBoxValue**) property to one of the Checked, Unchecked, or Grayed property values. Setting the cell's **CheckBoxValue** (**ItemCheckBoxValue** property) to any other value likewise sets the state to Grayed.

```
TList.Grid.Cells(2,3).CheckBoxValue = 0 'set to the default Unchecked value  
TList.ItemCheckBoxValue(15) = 1 'set to the default Checked value  
TList.Node(15).CheckBoxValue = 1 'set to the default Checked value
```

---

## Children Property (TListNode Object)

### Applies to

TListNode object

### Description

Returns a reference to a TListNode collection holding the subordinate Node objects. This property is read-only.

### Syntax

```
Node.Children
```

### Data Type

TListNodes

### Example

```
' Set FontColor for all children of some node  
Dim Children as TListNodes  
Set Children = TList.Node(l).Children  
For iChild = 0 to Children.count-1  
    Children.Item(iChild).celldef.ForeColor  
' or    Children.(iChild).celldef.ForeColor  
Next iChild  
Or  
' Set FontColor for all children of some node  
Dim Children as TListNodes
```

```
Set Children = TList.Node(l).Children
For each TListItem in Children
    TListItem.cellDef.ForeColor = vbBlue
Next
```

### [See Also](#)

**ChildrenCount** property

---

## ChildrenCount Property (TListNode Object)

### [Applies to](#)

**TListNode** object

### [Description](#)

Returns the number of subordinate child Node objects owned by the Node object. This property is read-only.

### [Syntax](#)

```
[Number&] = Node.ChildrenCount
```

### [Data Type](#)

Long

### [See Also](#)

**Children** property

---

## Col and Row Properties (TListGridCell object)

### [Applies To](#)

**TListGridCell** object

### [Description](#)

The **Row** and **Col** properties of a **TListGridCell** return the coordinates of the current cell in a Grid. Read-only. Not available at design time.

### [Syntax](#)

```
C& = TListGridCellObject.Col
R& = TListGridCelObject.Row
C& = [form.]TList.Grid.Cells(Row&, Col&).Col
R& = [form.]TList.Grid.Cells(Row&, Col&).Row
C& = [form.]TList.ItemGrid(ItemIndex).Cells(Row&, Col&).Col
R& = [form.]TList.ItemGrid(ItemIndex).Cells(Row&, Col&).Row
```

### [Remarks](#)

Use these properties to Identify which row or column contains the current cell. Columns and rows are numbered from zero, beginning at the top for rows and at the left for columns. To determine what Grid object has the focus use TList's **ActiveGrid** property.

---

## Col and Row Properties (TListGrid object)

### [Applies to](#)

**TListGrid** object

### [Description](#)

These properties return the ordinal number of the active cell/row in the active grid.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

### Applies to

TListGrid object

### Syntax

```
[long] = [TListGrid].Col  
[long] = [TListGrid].Row
```

### Remarks

Use these properties to identify which row or column contains the active cell.

Use the ActiveRow and ActiveCell properties to reference the ActiveRow or Cell.

Columns and rows are numbered from zero, beginning at the RowTitles column as column 0 and the Column Titles row as row 0 (even if the **ShowRowTitles** and/or **ShowColumnTitles** properties are false)

**Col** and **Row** properties can return -1 value if there is no active cell in the grid (other grid or item is active) or the entire row is selected.

These are Read-Only properties - Use the Activate method to activate a Cell or Row.

### Example

```
Value = TList.Grid.Cells( TList.Grid.Row, TList.Grid.Col).Value  
' Is equivalent to  
Value = TList.Grid.ActiveCell.Value
```

---

## ColDefs Property

### Applies To

TListGrid object

### Description

The **ColDefs** property returns a reference to the TListColDefs object that holds a series of **ColDef** objects. A **ColDef** object contains formatting information for a column.

---

TList 8 supports referencing TListColDefs and TListGridCells by ValueName as well as by column index.

---

### Syntax

```
[TListColDefs] = TListGridObject.ColDefs  
TListGridObject.ColDefs(1).CellDef.BackColor = RGB(255, 0, 0)  
or  
TListGridObject.ColDefs("ColumnName").CellDef.BackColor = RGB(255, 0, 0)
```

---

## ColDelimiter Property

### Applies to

TList object

### Description

Specifies a character, which will be used as a delimiter by the **AddRow** and **AddItem** methods to distinguish data for different cells entered as a single string.

```
TList1.ColDelimiter = 9 ' Use Tab as delimiter  
' This will create a 2 column Tree Grid  
TList1.AddItem "Data1"& Chr(9) & "Data2"
```

### Syntax

```
[form.]TList1.ColDelimiter [ = character_code%]
```

### Remarks

The *character\_code* is a number that identifies a character. The tabulation character (9) is the default setting for this property.

### Data Type

Integer

### See Also

**ConvertTabsToCols** and **TabStopDistance** properties

---

## Cols and Rows Properties

### Applies To

**TListGrid** object

### Description

To read or set the number of columns of a grid use the **Grid.Cols** property.

To read the number of rows of a grid use the **Grid.Rows** property.

### Syntax

```
TListGridObject.Cols = [ NumberOfCols&]  
R& = TListGridObject.Rows = [ NumberOfRows&]  
C& = [form.]TList.Grid.Cols = [ NumberOfCols&]  
R& = [form.]TList.Grid.Rows = [ NumberOfRows&]  
C& = [form.]TList.ItemGrid(ItemIndex).Cols = [ NumberOfCols&]  
R& = [form.]TList.ItemGrid(ItemIndex).Rows = [ NumberOfRows&]
```

---

## ColTitleCellDef Property

### Applies To

**TListGrid** object

### Description

Returns a reference to a **TListCellDef** object. This object collects all properties which determine the default formatting for grid column titles. This property doesn't affect formatting of either other grid cells or row titles.

### Syntax

```
[form.]TList1.Grid.ColTitleCellDef  
[form.]TList1.ItemGrid(ItemIndex&).ColTitleCellDef
```

### Example

```
TList1.Grid.ColTitleCellDef.BackColor = RGB(127, 127, 127)
```

---

## ColTitlesHeight Property

### Applies To

**TListGrid** object

### Description

Returns or sets the height of the grid row containing column titles expressed in twips.

Not available at design time.

### Syntax

```
TListGridObject.ColTitlesHeight [= number& ]  
[form.]TList.Grid.ColTitlesHeight [= number& ]  
[form.]TList.ItemGrid(ItemIndex&).ColTitlesHeight [= number& ]
```

## Data Type

Single

---

# ComboBox Property (TListEditInfo Object)

## Applies to

TListEditInfo object

## Description

This property returns a **TListComboBox** object controlling combobox editing for a data cell or cells  
Note: The EditInfo.**Style** property must be set to TLEDITINFO\_COMBOBOX before accessing the ComboBox property

## Syntax

```
[TListCellDef].EditInfo.ComboBox
```

## Data Type

TListComboBox object

---

# ConvertTabsToCols Property

## Applies to

TList object

## Description

Determines whether the **AddItem** method automatically creates columns if there is a **ColDelimiter** character in an input string.

## Syntax

```
[form.]TList.ConvertTabsToCols [= {True/False} ]
```

## Data Type

Integer

## See Also

ColDelimiter, TabStopDistance, and AddItem properties

---

# Count Property

## Applies To

TListValues Object

TListLevelDefs Object

TListColDefs Object

TListNodes object

## Description

Returns the number of items in an object collection.

For example, for **TListColDefs** object collection this property returns the number of **TListColDef** objects stored in this collection. Read-only.

## Syntax

```
[form.]TList1.LevelDefs.Count  
[form.]TList1.Grid.ColDefs.Count  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs.Count
```



```
[form.]TList1.ItemValues(ItemIndex&).Count
```

### Data Type

Long

---

## Count Property (TListSelectedGridCells, TListSelectedGridColumnns and TListSelectedGridRows Objects)

### Applies to

TListGrid object

### Description

This property returns the number of objects in a collection that holds a series of **TListGridCell**, **TListColDef** and **TListRowDef** objects representing all selected cells/columns or rows.

### Syntax

```
[long] = [TListGrid].SelectedCells.Count  
[long] = [TListGrid].SelectedColumns.Count  
[long] = [TListGrid].SelectedRows.Count
```

### Example

- Display a count of all selected cells

```
Dim objSelectedCells as TListSelectedGridCells  
Set objSelectedCells = TList1.Grid.SelectedCells  
Debug.Print "Selected cells count: " & objSelectedCells.Count
```

- Display a count of all selected rows

```
Dim objSelectedRows as TListSelectedGridRows  
Set objSelectedRows = TList1.Grid.SelectedRows  
Debug.Print "Selected rows count: " & objSelectedRows.Count
```

- Display a count of all selected columns

```
Dim objSelectedColumns as TListSelectedGridColumnns  
Set objSelectedColumns = TList1.Grid.SelectedColumns  
Debug.Print "Selected columns count: " & objSelectedColumns.Count
```

---

## Count Property (TVWSelectedNodes Object)

### Description

Returns the number of selected nodes in an SelectedNodes collection. Read-only.

### Syntax

```
[form.]TListTreeView.SelectedNodes.Count
```

### Data Type

Long

---

## Clear Method

### Applies to

TList object

### Description

Clears the list of items from TList.list

### Syntax

```
[form.]TList1.Clear
```

### Remarks

Calls to this method update the control unless the **Redraw** property is set to False.

The list of items affected by this method may be limited by the CurrentIndexMethod property. The entire tree is cleared only if **CurrentIndexMethod** is set to 0 or 1.

---

## Clear Method (TListNode Object)

### Applies to

TListNode object

### Description

Clears the list of children belonging TListNode Object

### Syntax

```
TListNode.Clear
```

### Remarks

Calls to this method update the control unless the **Redraw** property is set to False.

---

## Clear Method (TListSelectedGridCells, TListSelectedGridColumn and TListSelectedGridRows Objects)

### Applies to

TListGrid object

### Description

This method clears contents a collection that holds a series of **TListGridCell**, **TListColDef** and **TListRowDef** objects representing all selected cells/columns or rows. Deselects all cells, rows or columns of the grid object.

### Syntax

```
[TListGrid].SelectedCells.Clear()  
[TListGrid].SelectedColumns.Clear()  
[TListGrid].SelectedRows.Clear()
```

### Example

a) Clears all selected cells

```
Dim objSelectedCells as TListSelectedGridCells  
Set objSelectedCells = TList1.Grid.SelectedCells  
Call objSelectedCells.Clear()
```

b) Clears all selected rows

```
Dim objSelectedRows as TListSelectedGridRows  
Set objSelectedRows = TList1.Grid.SelectedRows  
Call objSelectedRows.Clear()
```

c) Clears all selected columns

```
Dim objSelectedColumns as TListSelectedGridColumnns
Set objSelectedColumns = TList1.Grid.SelectedColumns
Call objSelectedColumns.Clear()
```

---

## Clear Method (TListComboItems Object)

### Applies to

TListEditInfo object

### Description

This method removes all the items from the **TListComboItems** collection.

### Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.Clear
```

### Example

```
With TList.Grid.Coldefs(1).CellDef.EditInfo.ComboBox
'clear the combo list (it is not necessary to do for just created lists)
.Items.Clear
'add four items to the combo list with pictures
.Items.Add 10, Picture1.Picture, "Ten"
.Items.Add 20, Picture2.Picture, "Twenty"
.Items.Add 30, Picture3.Picture, "Thirty"
.Items.Add 40, Picture4.Picture, "Fourty"
'Replace the list box item "Fourty" with "Forty"
.Items.SafeAdd 40, Picture4.Picture, "Forty"
'add another item to the combo list, but with text above picture
Dim objComboItem as TListComboItem
Set objComboItem = .Items.Add(50, Picture5.Picture, "Fifty")
objComboItem.Alignment = TLALIGNMENT_PICTURE_BELOW_TEXT
End With
```

---

## Clear Method (TVWSelectedNodes Object)

### Description

Deselects all nodes.

### Syntax

```
[form.]TListTreeView.SelectedNodes.Clear
```

### Remarks

Note that clearing the selection does not remove the focus from the one node pointed to by SelectedItem. That node will no longer be selected but it will still have the focus and will still be pointed to by SelectedItem.

---

## ClearItem Property

### Applies to

TList object

### Description

Removes all subordinate items from an item specified by its *index*.

Not available at design time and write-only at run time.

### Syntax

```
[form.]TList1.ClearItem = index&
```

### Remarks

Setting this property updates the control unless the **Redraw** property is set to False.

### Data Type

Long

---

## Click Event

### Applies to

TList object

### Description

This event occurs when the user selects an item in TList control, either by pressing the arrow keys or by clicking the mouse button.

### Syntax

```
Sub TList_Click([Index As Integer])
```

### Remarks

For more information, see the description of the **Click** event in the *Microsoft Visual Basic On-Line Help*.

### See Also

*RightClick* event

---

## Clipboard Property

### Applies to

TList object

### Description

Setting the **Clipboard** property will copy or paste items to or from the Windows clipboard.

Index specifies the source or destination item upon which the actions will be done.

Not available at design-time, write-only at run-time.

### Syntax

```
[form.]TList1.Clipboard(index&)=enum%
```

### Settings

The **Clipboard** property settings are:

Setting	Description
0	Copy Item and children.
1	Copy item without its children.
2	Copy only children of the item.
3	Paste items from clipboard to the end of the list of the specified item's children
4	Paste items from clipboard before the specified item at the same indentation level.

### Remarks

TList's own format allows exchange of data, including any associated graphics and item data as well as the text, between TList controls via the clipboard. Such data is passed only between TList controls. TList also copies a text presentation of the copied tree. This can be passed to other applications.

if *Index* is set to a value of -1:

- If the **CurrentIndexMethod** property is set to 0 or 1, all items with an **Indent** property of 0 will be copied to the clipboard;
- If the **CurrentIndexMethod** property is set to 2 then all subordinates of the current item will be copied to the clipboard without their children.

---

Note

- **Tree Grid formatting (Number of columns, gridlines, show row titles, etc) is not copied to the clipboard.**
  - **The Clipboard property depends on the VisualRoot property settings.**
- 

### Data Type

Integer

---

## CoerceIndex Property

### Applies to

TList object

### Description

The **CoerceIndex** property accesses an index value in an internal buffer, converting that index as needed for a given indexing scheme.

Setting this property writes the index value to the buffer, Reading the property retrieves an index value.

Not available at design time.

*This property is preserved for the sake of backwards compatibility with older editions of TList. The TranslateIndexmethod should be used for translating between index methods.*

### See Also

The **CurrentIndexMethod** property, and **TranslateIndex** method

---

## Collapse Event

### Applies to

TList object

### Description

The **Collapse** event is generated whenever an item is collapsed, which means the item's subordinate items are not shown.

### Syntax

Sub TList_Collapse([Index As Integer,] / As Long)
---

### Remarks

This event passes *I*, the *index* of the item in the list that was collapsed.

---

## CopyBuffer Method

### Applies to

TList object

### Description

The **CopyBuffer** method copies a TList *tree buffer* to the clipboard.

The return value is zero if the method is successful. Otherwise, the return value is an error code.

Not available at design time.

---

You can use any TList control to copy any *tree buffer*.

---

### Syntax

```
[form.]TList1.CopyBuffer (ByVal hTreeBuffer&) As Integer
```

### Example

```
Dummy& = TList1.CopyBuffer(hTreeBuffer&)
```

---

## CopyItem Property

### Applies to

TList object

### Description

Upon reading this property, TList creates a *tree buffer* containing a copy of the specified item and its subordinate items. A long integer pointing to that *tree buffer* is returned.

Not available at design time and read-only at run time.

### Syntax

```
[form.]TList.CopyItem(index&)
```

### Remarks

Use **FreeBuffer** method to free memory when you are done with the *tree buffer*.

Use an Index of -1 to copy the entire tree.

---

### Note

- The Tree **Grid** formatting cannot be copied to a *tree buffer*.
  - The CopyItem property depends on the VisualRoot property settings.
- 

### Example

```
Dim tree_buffer&  
tree_buffer& = TList1.CopyItem(index&)  
...  
TList1.FreeBuffer(tree_buffer&)
```

### Data Type

Long

---

## CopyItemSub Property

### Applies to

TList object

### Description

Upon reading this property, TList creates a *tree buffer* containing a copy of all subordinates of the specified item. A long integer pointing to that *tree buffer* is returned.  
Not available at design time and read-only at run time.

### Syntax

```
[form.]TList.CopyItemSub(index&)
```

### Example

```
Dim tree_buffer&  
tree_buffer& = TList1.CopyItemSub(index&)  
...  
TList1.FreeBuffer(tree_buffer&)
```

### Remarks

If Index is set to a value of -1, Then

- If current index method is set to 0 or 1, all root items (items with **Indent** = 0) will be copied to the *tree buffer*.
- If current index method is set to 2, all subordinates of the current parent will be copied to the *tree buffer* without their children.

Use **FreeBuffer** method to free memory when you are done with the *tree buffer*.

If there are no subordinate items in the source item, then a **Nothing to copy** trappable error is generated.

---

Note: the CopyItemSub property depends on the VisualRoot property settings.

---

### Data Type

Long

---

## CopyOne Property

### Applies to

TList object

### Description

Upon reading this property, TList creates a *tree buffer* containing a copy of the specified item, not including any subordinates. A long integer pointing to that *tree buffer* is returned.

Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.CopyOne(Index&)
```

### Remarks

Use the **FreeBuffer** method to free memory when you are done with the *tree buffer*.

---

Note the **CopyOne** property depends on the **VisualRoot** property settings.

---

### Data Type

Long

---

## CopySelected Property

### Applies to

TList object

## Description

Upon reading this property, TList creates a *tree buffer* containing a copies of **selected** items. A long integer pointing to that *tree buffer* is returned.

Not available at design time and read-only at run time.

## Syntax

```
[form.]TList.CopySelected
```

## Remarks

Use **FreeBuffer** method to free memory when you are done with the *tree buffer*.

If there are no selected items in a list, then a **Nothing to copy** trappable error is generated.

The hierarchic shift is NOT preserved when copying selected items to a *tree buffer*

## Example

```
Dim tree_buffer&  
treebuffer& = TList1.CopySelected ' copy selected items to tree buffer  
TList2.Add(-1) = treeBuffer& ' add them to another TList control  
TList2.FreeBuffer(treebuffer&) ' free the tree buffer memory
```

## Data Type

Long

---

# CurrentIndexMethod Property

## Applies to

TList object

## Description

Specifies the method by which items in the list are enumerated. Setting this property determines how an index, used with other properties, is interpreted.

## Syntax

```
[form.]TList.CurrentIndexMethod[=enum_expr%]
```

## Settings

Settings for the **CurrentIndexMethod** property are:

Constant	Setting	Description
TLSYS_ENUM	0	(Default). Specifies enumeration includes all items in a list.
TLSYS_VIS	1	Specifies enumeration includes all visible items in a list.
TLSYS_LEVEL	2	Specifies enumeration includes all children of the <i>current parent</i> .

## Remarks

There are three possible methods to enumerate items in the list:

1. Disregarding each item's visibility, TList enumerates all items in the list. TList sequentially numbers each item according to its position in the list (distance from the top), starting with zero.
2. TList Enumerates sequentially ONLY VISIBLE items (items which may be seen by scrolling within the list - without expanding any parents not already expanded).
3. Access Items using path in a manner analogous to the MS-DOS file structure, with an index enumerating only immediately subordinate items.

For a detailed description of these methods refer to the introductory notes regarding selecting an Indexing scheme.



To convert an index from one method to another, you can use the **TranslateIndex** method.

---

Note

- Changes in the **CurrentIndexMethod** property may result in changes to the **ListCount** and **ListCountEx** properties
  - The **CurrentIndexMethod** property does not depend on the **VisualRoot** property settings.
- 

### Data Type

Integer

---

## CurrentItem Property

### Applies to

TList object

### Description

This property is obsolete even though it is still functional, but it might not be supported in future versions of TList. Please use **CurrentParent** property instead. The **CurrentParent** property provides exactly the same functionality as **CurrentItem** did in past editions of TList.

---

## CurrentParent Property

### Applies to

TList object

### Description

Specifies the *current parent*. The current parent is the parent whose children are enumerated in the list when using **CurrentIndexMethod** TLSYS\_LEVEL (when **CurrentIndexMethod** property is set to 2).

Not available at design time.

### Syntax

`[form.]TList.CurrentParent[ = Variant]`

### Settings

The **CurrentParent** property settings are:

Setting	Description
"\"	(Default)Makes the <i>current parent</i> the root
"."	Identifies a <i>current parent</i> .
"Item2"	Make "Item2" the <i>current parent</i> .
".."	Go one level up.
"<pound sign>12"	Make item with index 12 the <i>current parent</i> .
long	The index of an item which becomes the <i>current parent</i> .

### Remarks

Use the **PathSeparator** property to create a delimiter between the components of the **CurrentParent** property.

---

Note that the **CurrentParent** property is only of use when the **CurrentIndexMethod** = TLSys\_Level (2).

---

**CurrentParent** does **NOT** refer to the currently selected item (which is specified by **ListIndex**). In fact **CurrentParent** is always the parent of the item referenced by the **ListIndex** property. Any action changing the **ListIndex** value may change the value of **CurrentParent**. Properties of the **CurrentParent** may be set by using the special index value "-2":

```
TList1.CurrentIndexMethod = 2  
TList1.List(-2) = "New Current Parent Text"
```

### Example

```
TList1.CurrentParent = "..\Item23\Item27"  
TList1.CurrentParent = "..\..\..\Item23\Item27"  
TList1.CurrentParent = "Item23\Item27"  
TList1.CurrentParent = "\Item23\Item27"  
TList1.CurrentParent = "\Item23\Item27\#23\#0\#12"  
TList1.CurrentParent = 25
```

### Data Type

Variant

---

## CurrentItemBM Property

### Applies to

TList object

### Description

The **CurrentItemBM** property may be used to create or move to a bookmark. Reading **CurrentItemBM** returns the bookmark of the current parent. Not available at design time.

### Syntax

```
[form.]TList.CurrentItemBM [= bookmark&]
```

### Data Type

Long

### Remarks

Setting **CurrentItemBM** changes the *current parent* (not the selected item) to the item specified by a bookmark value. If you set this property to zero (0&) the root will be the *current parent*. If **CurrentItemBM** is set to -1 and the **CurrentIndexMethod** property is 1 **CurrentItemBM** will refer to **VisualRoot**.

### See Also

The **CurrentParent** property

---

## DateTime Property (TListEditInfo Object)

### Applies to

TListCellDef object

### Description

This property returns the **TListDateTime** object used for formatting, representing and editing date/time formatted data. The properties of the DateTime object support setting the data format, maximum and minimum values, etc.

Note: the **Style** property of the **EditInfo** object must be set to TLEDITINFO\_DATETIME before accessing the DateTime property.

## Syntax

```
Set objDate = [TListCellDef].EditInfo.DateTime
```

## Data Type

TListDateTime object

---

# DefItemCellAlignment and Alignment Properties

## Applies To

**DefItemCellAlignment** applies to TList Control.

**Alignment** applies to TListCellDef object

## Description

Each TList grid cell may contain both text and an image.

The **Alignment** property determines the alignment between text and picture in a cell. The

**DefItemCellAlignment** determines the default alignment of all items in the grid.

## Syntax

```
[form.]TList1.Grid.Cells(R, C).CellDef.Alignment = [enum%]  
[form.]TList1.DefItemCellAlignment = [enum%]
```

## Settings

The **Alignment** and **DefItemCellAlignment** properties settings are:

Setting	Value	Description
	-1	(Default for <b>CellDef.Alignment</b> property) Use default settings. This setting is not valid for <b>DefItemCellAlignment</b> property.
TLALIGNMENT_PICTURE_LEFT_OF_TEXT	0	(Default for <b>DefItemCellAlignment</b> property) Picture is aligned to the left of the object's text.
TLALIGNMENT_PICTURE_RIGHT_OF_TEXT	1	Picture is aligned to the right of the object's text.
TLALIGNMENT_PICTURE_ABOVE_TEXT	2	Picture is aligned above the object's text.
TLALIGNMENT_PICTURE_BELOW_TEXT	3	Picture is aligned below the object's text.

## Remarks

Setting of **Alignment** and **DefItemCellAlignment** properties updates control, unless the **Redraw** property is set to False.

# DefItemCellPictureAlignment and PictureAlignment Properties

## Applies To

**DefItemCellPictureAlignment** applies to TList Control.

**PictureAlignment** applies to TListCellDef object

## Description

**DefItemCellPictureAlignment** specifies the default picture alignment for all items of a control.

**PictureAlignment** property specifies how the picture will be displayed in a cell.

## Syntax

```
[form.]TList1.Grid.Cells(R, C).CellDef.PictureAlignment = [Enum%]  
[form.]TList1.DefItemCellPictureAlignment = [Enum%]
```

## Settings

The **DefItemCellPictureAlignment** and **PictureAlignment** property settings are:

Setting	Value	Description
	-1	(Default for <b>PictureAlignment</b> property) Use default settings. This setting is not valid for <b>DefItemCellPictureAlignment</b> property.
TLPICTUREALIGNMENT_NOT_VISIBLE	0	Picture is not displayed.
TLPICTUREALIGNMENT_LEFT_TOP	1	(Default for <b>DefItemCellPictureAlignment</b> property) Image is left/top aligned.
TLPICTUREALIGNMENT_LEFT_MIDDLE	2	Image is left/middle aligned.
TLPICTUREALIGNMENT_LEFT_BOTTOM	3	Image is left/bottom aligned.
TLPICTUREALIGNMENT_CENTER_TOP	4	Image is center/top aligned.
TLPICTUREALIGNMENT_CENTER_MIDDLE	5	Image is center/middle aligned.
TLPICTUREALIGNMENT_CENTER_BOTTOM	6	Image is center/bottom aligned.
TLPICTUREALIGNMENT_RIGHT_TOP	7	Image is right/top aligned.
TLPICTUREALIGNMENT_RIGHT_MIDDLE	8	Image is right/middle aligned.
TLPICTUREALIGNMENT_RIGHT_BOTTOM	9	Image is right/bottom aligned.
TLPICTUREALIGNMENT_STRETCH	10	Image is stretched to fit image area.
TLPICTUREALIGNMENT_COUPLED	11	Image is coupled with the text so these two objects can be handled like one complex object via <b>Alignment</b> and <b>TextAlignment</b> properties. If this flag is specified there will be no gap between picture and text.
TLPICTUREALIGNMENT_TILE	12	Image is tiled

---

**Note:** In order to center a picture inside a cell while there is no text in this cell it is necessary to set **TextAlignment** property to **TLTEXTALIGNMENT\_NOT\_VISIBLE**

---

## DefItemCellTextAlignment and TextAlignment Properties

### Applies To

**DefItemCellTextAlignment** applies to TList Control.

**TextAlignment** applies to TListCellDef object

### Description

**DefItemCellTextAlignment** specifies the default text alignment for all items of a control.

**TextAlignment** specifies the text alignment for a cell.

### Syntax

```
[form.]TList1.Grid.Cells(R, C).CellDef.TextAlignment = [enum%]  
[form.]TList1.DefItemCellTextAlignment = [enum%]
```

### Settings

The **DefItemCellTextAlignment** and **TextAlignment** properties settings are:

Setting	Value	Description
	-1	(Default for <b>TextAlignment</b> property). This setting is not valid for DefTextAlignment property.
TLTEXTALIGNMENT_NOT_VISIBLE	0	Text is not displayed.
TLTEXTALIGNMENT_LEFT_TOP	1	Text is left/top aligned.
TLTEXTALIGNMENT_LEFT_MIDDLE	2	Text is left/middle aligned.
TLTEXTALIGNMENT_LEFT_BOTTOM	3	Text is left/bottom aligned.
TLTEXTALIGNMENT_CENTER_TOP	4	Text is center/top aligned.
TLTEXTALIGNMENT_CENTER_MIDDLE	5	Text is center/middle aligned.
TLTEXTALIGNMENT_CENTER_BOTTOM	6	Text is center/bottom aligned.
TLTEXTALIGNMENT_RIGHT_TOP	7	Text is right/top aligned.
TLTEXTALIGNMENT_RIGHT_MIDDLE	8	Text is right/middle aligned.
TLTEXTALIGNMENT_RIGHT_BOTTOM	9	Text is right/bottom aligned.

---

## DbClick Event

### Applies to

TList object

### Description

This event occurs when the user double-clicks an item in a TList control.

### Syntax

```
Sub TList_DbClick([Index As Integer])
```

### Remarks

For more information, see the description of the **DbClick** event in the *Microsoft Visual Basic On-Line Help*.

---

## DefItemCellDef Property

### Applies to

TList object

### Description

This property returns a reference to a TListItemCellDef object. This object is used to collect all properties, which determine how an item cell will be displayed by default. This property affects all items in the tree.

### Syntax

```
[form.]TList1.DefItemCellDef
```

### Example

```
TList1.DefItemCellDef.BackColor = RGB(127, 127, 127)
```

---

## DefMultiLine Property

### Applies to

TList object

### Description

Determines whether TList defaults to wordwrapping long text items, or keeping text on a single line.

### Syntax

```
[form.]TList.DefMultiLine[=boolean %]
```

### Settings

The **DefMultiLine** property settings are:

Setting	Description
False	(Default) TList's default behavior is to ignore carriage returns and restrict data to a single line.
True	TList will default to wordwrapping long lines of text.

### Remarks

This property may be overridden for individual items by setting of the **ItemMultiLine** property to True.

### Data Type

Boolean

---

## DisableNoScroll Property

### Applies to

TList object

### Description

By default, scrollbars are not shown unless required. Setting this property to True shows the disabled scrollbars, otherwise the scroll bars are hidden. Read only at run-time.

### Syntax

```
[form.]TList.DisableNoScroll
```

### Settings

The **DisableNoScroll** property settings are:

Setting	Description
True	Show disabled scroll bars.
False	(Default) Don't show disabled scroll bars.

### Remarks

The **Scrollbars** property is used to enable horizontal and/or vertical scrollbars. The **DisableNoScroll** simply determines how to handle the enabled scrollbar when there is nothing to scroll.

### Data Type

Boolean

---

## DisplayValue Property (TListComboItem Object)

### Applies to

TListComboBox object

### Description

The **DisplayValue** property specifies the text shown for each item in in the drop down portion of a combobox during in-place editing. As the user selects items from the list the display value for the selected item will also be shown within the text entry area of the combobox. When editing is completed the DisplayValue will be shown in the edited cell. If no DisplayValue is set the CellValue property is used.

The DisplayValue property is also used to support intelligent formatting – for cells with associated combobox editing objects TList compares the Value of the cell with each the CellValue property of each combobox list item. When a match is found the DisplayValue is shown in that cell along and formatted according to the specifications of color, font, alignment and display picture for the combobox list item.

The value of this property is identical to the 3<sup>RD</sup> parameter in the **Add** method of the **TListComboItems** object. It may also be set or read directly.

### Syntax

```
[TListCombobox].Items(Index).DisplayValue = [Data to display]
```

### Example

```
1) Specify use of a combobox for editing column 5 of a TList grid
   TList1.Grid.Coldefs(5).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCLICK
   TList1.Grid.Coldefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
'Set up the combobox list items
With TList1.Grid.ColDefs(5).CellDef.EditInfo.ComboBox.Items
  'the 3rd parameter of the Combobox item's ADD method sets the DisplayValue
  .Add 1, , "1 Hour" ' user sees "Hour" representing Value of 1
  ' alternatively the Display Value may be set separately
  .Add 24
  .Item( .count-1 ).DisplayValue = "1 Day"
  .Add 168
  .Item( .count-1 ).DisplayValue = "1 Week"
End With
The user may now select an item from the combobox. Selecting the second item sets the cell's value to 24 but the text
displayed in the cell will be "1 Day"
```

```

2) Note that the Combobox may be used to control the display text even without enabling editing
TList1.Grid.ColDefs(5).CellDef.EditInfo.Editable = TLEDITABLE_NEVER ' don't allow editing
TList1.Grid.Cells( 2, 5 ).value = 168 ' after assigning a value, TList sets the display
                                     ' text based on matching the Cell's value to the
                                     ' CellValue of the combobox list items
The user will now see the string "1 Week" in row 2, column 5

```

## DisplayPic Property (TListComboItem Object)

### Applies to

TListComboBox object

### Description

The **DisplayPic** property of each **TListComboItem** object determines the picture shown for that combobox list item in the drop down portion of the combobox.

For intelligent formatting – if a **TListCombobox** object is associated with a cell (even without enabling editing), TList compares the cell's value with the **CellValue** properties of the combobox list items and displays the corresponding display picture in the cell.

The value of this property is set using the 2<sup>RD</sup> parameter in the **Add** method of the **TListComboItems** object.

### Syntax

```
[TListCombobox].Items(Index).DisplayPic = [Picture]
```

### Example

```

1) Specify use of a combobox for editing column 5 of a TList grid
TList1.Grid.ColDefs(5).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCLICK
TList1.Grid.ColDefs(5).CellDef.EditInfo.Style = TLEDITINFO_COMBOBOX
'Set up the combobox list items
With TList1.Grid.ColDefs(5).CellDef.EditInfo.ComboBox.Items
  'the 2nd parameter of the Combobox items' add method sets the DisplayPicture
  .Add 1, TList1.LoadPicture("apples.ico"), "Apples"
  .Add 2, TList1.LoadPicture("Oranges.ico"), "Oranges"
  .Add 3
  .Item( .Count - 1).DisplayPicture = Picture1.Picture
  .Item( .Count - 1).DisplayValue = "Grapes"
End With
'Optionally hide the CellValues so you only see the pictures
TList1.Grid.ColDefs(5).CellDef.TextAlignment = TLTEXTALIGNMENT_NOT_VISIBLE

```

The user may now select an item from the combobox. Selecting the second item sets the cell's value to 1 and displays a picture of an Apple in the cell.

```

2) Even without editing the DisplayPic may be used for intelligent formatting
TList1.Grid.ColDefs(5).CellDef.EditInfo.Editable = TLEDITABLE_NEVER
TList1.Grid.Cells( 2, 5 ).value = 1 ' after assigning a value, TList sets the display
                                     ' picture based on matching the Cell's value to the
                                     ' CellValue of the combobox list items
The user will now see the picture associated with a value of 3 – presumably looks like an grape.

```

### Remarks



If the **DisplayPic** property is not set then the cell's picture ( as set by the **.CellPicture** property of the cell) will be displayed instead.

---

## Drag Method

### Applies to

TList object

### Description

Begins, ends, or cancels dragging TList controls.

### Syntax

```
[form.]TList.Drag [,action%]
```

### Remarks

For more information, see the description of the **Drag** method in the *Microsoft Visual Basic On-Line Help*.

You should call **BeforeDrag** method before starting the Drag Drop operations:

```
TList1.BeforeDrag  
TList1.Drag 1
```

---

## DragColumnsEnabled property

### Applies to

TListGrid object

### Description

The **DragColumnsEnabled** property enables or disables the end-user ability to drag columns of a grid with the mouse. When enabled the user can click and drag on a grid column header to drag the data from one column to another.

### Syntax

```
TList.GridObject.DragColumnsEnabled [= bool%]
```

### Example

```
'Enable Column Dragging for TList Grid  
TList1.Grid.DragColumnsEnabled = True  
  
'Enable Column Dragging within a TList ItemGrid  
TList1.ItemGrid(1).DragColumnsEnabled = True  
TList1.Grid.DragColumnsEnabled = True
```

### Data Type

Boolean

### Remarks

The column can be moved only inside the grid object it belongs to.

---

## DragDrop, DragOver Events

### Applies to

TList object

### Description

See the Visual Basic™ *Language Reference* or Help for documentation on these events.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Syntax

```
Sub TList_DragDrop([Index As Integer,] Source As Control,  
    ➡ X As Single, Y As Single)  
Sub TList_DragOver([Index As Integer,] Source As Control,  
    ➡ X As Single, Y As Single, State As Integer)
```

## Remarks

Use the **DropTarget** property to get the *index* of the item which is currently under the mouse cursor. See *Drag Drop* for more details.

You MUST call **OnDragDrop** method as the first line in the **DragDrop** event and **OnDragOver** method as the first line in the **OnDragOver** event for any application involving an OCX edition of TList in Drag Drop:

```
Private Sub TList1_DragOver(Source As Control, _  
    X As Single, Y As Single, State As Integer)  
  
    TList1.OnDragOver X, Y, State  
    ...  
End Sub  
  
Private Sub TList1_DragDrop(Source As Control, _  
    X As Single, Y As Single)  
  
    TList1.OnDragDrop X, Y  
    ...  
End Sub
```

---

# DragDropEx, DragOverEx Events

## Applies to

TList object

## Description

These events are fired for TList as a drop target immediately before OleDragDrop and OleDragOver events when using TList's AutoDragDrop support.

## Syntax

```
Sub TList_DragDropEx(Source As TList, X As Single, Y As Single)  
Sub TList_DragOverEx(Source As TList, X As Single, Y As Single, State As Integer)
```

## Remarks

- Note these events are provided for migration of applications built prior to TList 8. In previous versions of TList AutoDragDrop support was based on Visual Basic Drag Drop technology and triggered VB style DragDrop and DragOver events. TList 8 now uses OLEDragDrop technology for internal drag drop mechanisms. As a result the VB events DragDrop and DragOver can not be triggered by the Automated drag drop actions. Old DragDrop code should therefore be moved from DragDrop and DragOver events to DragDropEx and DragOverEx events. New applications should simply use the OleDragDrop and OleDragOver events.
- The VB native DragDrop and DragOver events will still be triggered by drag and drop initiated programmatically ( ie: when calling the Drag method to initiate Drag and drop ). In this case the DragDropEx and DragOverEx methods will not be called.
- It is NOT necessary to call the OnDragDrop method or the OnDragOver methods within OnDragDropEX or OnDragOverEX events, but calling these methods will not cause a problem.

---

# DragHighlight Property

## Applies to

**TList** object

### Description

Determines whether and how items in the control will be highlighted when another control drags over it.

### Syntax

```
[form.]TList.DragHighlight[ = enum%]
```

### Settings

The **DragHighlight** property settings are:

Setting	Description
-1 ( <u>True</u> )	(Default) Items will be highlighted/ Inverted
0	Items won't be highlighted
1	Same as -1
2	Rectangle will be drawn around item.

### Data Type

Integer

---

## DragIcon Property

### Applies to

**TList** object

### Description

Determines the icon to be displayed as the pointer in a drag-and-drop operation.

### Syntax

```
[form.]TList.DragIcon [= icon]
```

### Remarks

The **DragIcon** property settings are:

Setting	Description
(none)	(Default) An arrow pointer inside a rectangle.
Icon	A custom mouse pointer. You specify the icon by loading it using the Properties window at design time. You can also use the <b>LoadPicture</b> function at run time. The file you load must have the .ICO file-name extension and format.

For more information, see the description of the **DragIcon** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Variant

---

## DragIconStyle Property

### Applies to

**TList** object

### Description

Determines the appearance of icon to be displayed as drag-and-drop operation pointer depending on the TList item being dragged.

### Syntax

```
[form.]TList.DragIconStyle [= enum%]
```

### Settings

The **DragIconStyle** property settings are:

Setting	Description
0	(Default) Use DragIcon picture.
1	Use standard drag/drop icon for all operations
2	Use smart icon. The icon appearance depends on the type of item being dragged. There are three types of icon: single item, item with children and selection of items

### Data Type

Long

---

## DragMode Property

### Applies to

TList object

### Description

Specifies use of either manual or automatic dragging mode for a drag-and-drop operations.

### Syntax

```
[form.]TList.DragMode [= enum%]
```

### Remarks

The **DragMode** property settings are:

Setting	Description
0	(Default) Manual; requires using the <b>Drag</b> method to initiate dragging on the source control.
1	Automatic; clicking the source control automatically initiates dragging.

For more information, see the description of the **DragMode** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Integer

---

## DrawFocusRect Property

### Applies to

TList object

### Description

Specifies whether to draw focus rectangle around the item pointed to by the **ListIndex** property (the item with the current focus).

### Syntax

[form.]TList.DrawFocusRect [= bool%]

### Remarks

The **DrawFocusRect** property settings are:

Setting	Description
True	(Default) Focus is drawn around selected item.
False	Focus is not drawn around selected item.

### Data Type

Boolean

---

## DropDownItemHeight Property (TListComboBox)

### Applies to

TListEditInfo object

TlistComboBox object

### Description

This property defines the height of list items in the **TListComboBox** and is set in pixels.

### Syntax

[TListEditInfo].EditInfo.ComboBox.DropDownItemHeight = [pixels%]

### Values

The **DropDownItemHeight** property settings are:

Setting	Description
-2	The height of each combobox list item is independent of the cell size, and determined by the item's settings (font, alignment, picture, etc) Items in the combobox may potentially have different heights; in this case the last item in the list may be only partially displayed.
-1	(Default) – The height of combobox list items is equal to the height of the cell being edited. All items in the combobox are the same height; the dropdown height will be adjusted to show only fully visible items.
> 0	This value specifies the height in pixels of all combobox list items. All items in the combobox list are the same height; the dropdown height will be adjusted to show only fully visible items.

---

## DropDownMaxHeight Property (TListComboBox)

### Applies to

TListComboBox object

### Description

This property sets the maximum height of the **TListComboBox** drop-down list in pixels.

### Syntax

CellDef.EditInfo.ComboBox.DropDownMaxHeight [= pix%]

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

### Values

The **DropDownMaxHeight** property settings are:

Setting	Description
-1	(Default) – the maximum height of the drop-down portion of <b>TListComboBox</b> is calculated automatically depending on the number and size of the items in drop-down list. Usually the maximum height is adjusted to show 6 items or less.
> 0	The maximum height of the drop-down portion of <b>TListComboBox</b> is specified by this property in pixels.

### Remarks

A scrollbar will be shown if the dropdown height is not tall enough to display all items.

The actual height of the drop down list is limited to the sum of heights of all items in the list

If **DropDownItemHeight** is set to -1 or > 0 (if all items are the same height) the actual height of the drop down list will be automatically decreased to display only fully visible items which can be shown within the **DropDownMaxHeight** value.

---

## DropDownWidth Property (TListComboBox)

### Applies to

**TListComboBox** object

### Description

This property sets the width of the **TListComboBox** drop-down list.

### Syntax

CellDef.EditInfo.ComboBox.DropDownWidth [= pix%]
--

### Values

The **DropDownWidth** property settings are:

Setting	Description
-2	The width of the drop-down portion of <b>TListComboBox</b> is automatically adjusted to completely show items in the list.
-1	(Default) – the width of the drop-down portion of <b>TListComboBox</b> is equal to the editing portion.
> 0	The width of the drop-down portion of <b>TListComboBox</b> is specified by this property in pixels.

---

## DropTarget Property

### Applies to

**TList** object

### Description

This property returns the *index* of the item under the mouse cursor during drag-and-drop operation, and returns -1 otherwise.

Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.DropTarget
```

### Data Type

Long

---

## EditAreaMinHeight, EditAreaMaxHeight, EditAreaMinWidth, EditAreaMaxWidth Properties (TListComboBox)

### Applies to

TListComboBox object

### Description

These properties may be used to set minimum and maximum height and width of the editing window in pixels independent of the height and width of the edited grid or item cell.

By default (with these properties set to -1) the size of the editing window is defined by the size of the grid or the item cell being edited.

### Syntax

```
CellDef.EditInfo.ComboBox.EditAreaMinHeight [= pix%]  
CellDef.EditInfo.ComboBox.EditAreaMaxHeight [= pix%]  
CellDef.EditInfo.ComboBox.EditAreaMinWidth [= pix%]  
CellDef.EditInfo.ComboBox.EditAreaMaxWidth [= pix%]
```

### Values

The **EditAreaXXXWidth** and **EditAreaXXXHeight** properties settings are:

Setting	Description
-1	(Default) – the height and width of the editing window depend on the size of the edited grid cell or item cell.
> 0	The height and width of the editing window - specified by this property in pixels.

---

## Editable Property(TListEditInfo)

### Applies to

TListEditInfo object

### Description

This property specifies determines whether a cell is editable

This property overrides EditingMode property settings and ItemEditText and CellEdit properties.

### Syntax

```
[TListCellDef].EditInfo.Editable = [enum%]
```

### Property Settings

Constant	Setting	Description
TLEDITABLE_DEFAULT	0	Whether or not the cell will be editable depends on <b>EditingMode</b> property flags or can be controlled by <b>ItemEditText</b> and <b>CellEdit</b> properties.
TLEDITABLE_NEVER	1	The cell can't be edited using in-place editors regardless of the settings of the <b>EditingMode</b> property. This setting also prevents editing using <b>CellEdit</b> and <b>ItemEditText</b> properties. Note: The value in this cell can still be set/modified programmatically.
TLEDITABLE_ALWAYS_ONCLICK	2	The cell is always editable; editing is automatically started as soon as user clicks over the cell.
TLEDITABLE_ALWAYS_ONDBLCLICK	3	The cell is always editable; editing is automatically started as soon as user double clicks over the cell.
TLEDITABLE_ALWAYS_ONACTIVATE	4	This setting indicates that TList should put the activated cell into edit mode as soon as the cell is activated This works as if user has the following code inside the GridCellActivate event <b><i>objGridCell.Grid.CellEdit(Row,Col) = TLEDITMODE_BEGIN</i></b>
TLEDITABLE_MULTIPLESTARTS	5	This setting indicates cell editing should be started as indicated by the settings of another property EditableStartOptions belonging to TListEditInfo object.

### Example

```
'Specify automatic editing for all cells in the grid (excluding captions) on double click
TList1.EditingMode = EDITMODE_GRID_CELLS
'disable editing all cells that belong to the third column
TList1.Grid.ColDefs(3).CellDef.EditInfo.Editable = TLEDITABLE_NEVER
'....
'enable editing on single click for second column
TList1.Grid.ColDefs(2).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCLICK
'enable editing on double click for fourth column
TList1.Grid.ColDefs(4).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCDBLCLICK
'....
'enable editing on single click for second column but disable editing a specific cell in this column
TList1.Grid.ColDefs(2).CellDef.EditInfo.Editable = TLEDITABLE_ALWAYS_ONCLICK
TList1.Grid.Cells(5, 2).CellDef.EditInfo.Editable = TLEDITABLE_NEVER
```

## EditableStartOptions Property(TListEditInfo)

### Applies to

TListEditInfo object

### Syntax

```
CellDef.EditInfo.EditableStartOptions = [enum%]
```



## Property Settings

Constant	Setting	Description
TL_EDITABLEOPTION_ON_CLICK	0	Whether or not the cell will be editable depends on <b>EditingMode</b> property flags or can be controlled by <b>ItemEditText</b> and <b>CellEdit</b> properties.
TL_EDITABLEOPTION_ON_DBLCLICK	1	The cell can't be edited using in-place editors regardless of the settings of the <b>EditingMode</b> property. This setting also prevents editing using <b>CellEdit</b> and <b>ItemEditText</b> properties. Note: the value in this cell can still be modified programmatically.
TL_EDITABLEOPTION_ON_ACTIVATE	2	The cell is always editable; editing is automatically started as soon as user clicks over the cell.

## Remarks

See also KeyboardNavigateWhileEditing property which can be used to automatically continue editing at the next cell in response to navigation during editing

---

# EditInfo Property

## Applies to

TListCellDef objects

## Description

This property references an *object* holding the editing style and providing access to the edit objects.

## Syntax

```
[TListCellDef].EditInfo
```

## Example

Setting the Style property of the EditInfo object to *TLEDITINFO\_TEXTBOX* instructs TList to use the TListTextBox object to for end-user editing.

```
With TList1.Grid.ColDefs(2).CellDef
.EditInfo.Style = TLEDITINFO_TEXTBOX
.EditInfo.TextBox.Options = TLTEXTBOX_OPT_AUTOWIDTH
.EditInfo.TextBox.MinWidth = 150      'min width of edit window in pixels
.EditInfo.TextBox.MaxWidth = 250     'max width of edit window in pixels
.EditInfo.TextBox.MaxLength = 10     'max number of characters in edit window
End With
```

## Data Type

TListEditInfo object

## Remarks

To reset the Editing style to none – set the EditInfo property to “Nothing”

```
Set TList1.Grid.ColDefs(1).CellDef.EditInfo = Nothing
```

---

## EditingKeyDown Event

### Applies to

TList object

### Description

This event is triggered as a result of a **KeyDown** during Editing.

### Syntax

```
Sub TList1_EditingKeyDown (ItemIndex As Long, KeyCode As Integer, Shift As Integer)
```

### Remarks

KeyCode and Shift parameters are as for the standard **KeyDown** event. ItemIndex refers to the item being edited.

---

## EditingKeyPress Event

### Applies to

TList object

### Description

This event is triggered as a result of a **KeyPress** during Editing.

### Syntax

```
Sub TList1_EditingKeyPress([Index As Integer,] ItemIndex As Long,  
    ➡ KeyAscii As Integer)
```

### Remarks

The KeyAscii parameter is as for the standard **KeyPress** event. ItemIndex refers to the item being edited.

---

## EditingKeyUp Event

### Applies to

TList object

### Description

This event is triggered as a result of a **KeyUp** during Editing.

### Syntax

```
Sub TList1_EditingKeyUp (ItemIndex As Long, KeyCode As Integer, Shift As Integer)
```

### Remarks

KeyCode and Shift parameters are as for the standard **KeyUp** event. ItemIndex refers to the item being edited.

---

## EditingMode Property

### Applies to

TList object

### Description

This property provides control over automatic cell and item editing process. If this property turned on, users will be able to start editing process by mouse clicking or double clicking over the cell or

item area. Editing is then concluded when the end user clicks on another control, or anywhere outside the edited item.

It is not necessary to write any VB code controlling the editing process (except if the developer wants to add some special data checking).

### Syntax

```
TList.EditingMode = Long
```

### Data Type

Long

### Settings

This is a BitFlag property. Values for setting this property are built by OR'ing the desired bit value constants:

Setting	Value	Description
EDITMODE_NONE	&H0	Disable automatic editing
EDITMODE_TREE_ITEMS	&H01	Enable automatic editing of tree items.
EDITMODE_GRID_CELLS	&H02	Enable automatic editing of all grid cells (except row and column captions).
EDITMODE_GRID_ROW_CAPTIONS	&H04	Enable automatic editing of grid row captions.
EDITMODE_GRID_COL_CAPTIONS	&H08	Enable automatic editing of grid column captions
EDITMODE_START_ON_CLICK	&H10	Start Editing in response to a single mouse click (note: by default a double click is required to start editing ).

### Remarks

The EditingMode property may be overridden for specified rows, columns, or grid cells, by the setting of the celldef.editinfo.**Editable** property.

### See Also

RequestEditing, GridCellRequestEditing, AfterEditing, GridCellAfterEditing, EditingKeyDown, GridCellEditingKeyDown, EditingKeyUp, GridCellEditingKeyUp, EditingKeyPress, and GridCellEditingKeyPress, events

CellEdit, ItemEditText and Editable properties

---

## EditInfoObject Property (TListEditingChangeInfo object)

### Applies to

TListEditingChangeInfo object

### Description

This property returns a reference to the **TListEditInfo** object being used for editing the current Tree or List Item, or Grid Cell.

### Syntax

```
[TListEditingChangeInfo].EditInfoObject = [TListEditInfo]
```

### Example

```

Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _
    ByVal objGridCell As TListGridCell, _
    ByVal objChangeInfo As TListEditingChangeInfo)

    ' update a textbox on a form during editing
    ' as user moves through items in editing combobox

    If objChangeInfo.ValueType = TLED_CHANGE_COMBOBOX_LISTINDEX
        'get a reference to the combobox item that was selected
        Dim objComboltem as TListComboltem
        Set objComboltem =
            objChangeInfo.EditInfoObject.ComboBox.Items(objChangeInfo.Value)
        'update outside TextBox object to current combobox list selection
        Text1.Text = objComboltem.CellValue
        Text1.BackColor = objComboltem.BackColor
    End If
End Sub

```

### Remarks

Note: This reference to the **TListEditInfo** object should be used primarily for getting the information about the edit object. It can potentially be used to change settings of the edit objects as well, but all such changes performed inside **GridCellEditingChange** and **ItemEditingChange** events will be applied to the edit object only after the data editing for that cell or item is completed or cancelled.

### See Also

**GridCellEditingChange** and **ItemEditingChange** events, **ValueType**, **Value**, **OldValue**, **SelStart**, **SelLength**, **EditInfoObject** properties

## Enabled Property

### Applies to

**TList** object

### Description

Determines whether the control is able to be acted upon.

### Syntax

```
[form.]TList.Enabled [= boolean%]
```

### Remarks

The **Enabled** property settings are:

Setting	Description
True	(Default) Allows the TList to respond to events.
False	Prevents the TList from responding to events.

For more information, see the description of the **Enabled** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Boolean

## EndPage Event

### Applies to

## TList object

### Description

This event is generated during the printing process just after TList prints the page. The user can handle this event to print any additional information on printer DC (for example page footer).

### Syntax

```
Sub TList_EndPage([Index As Integer,] PrinterObject As Variant, ReportPage As TListReportPage)
```

### Remarks

The **EndPage** event syntax has these parts:

Part	Description
Index	An integer that uniquely identifies a control if it is in a control array.
PictureObject	An output device, can be a printer object, a picture-box, or a DC handle
ReportPage	An object that describes the current page, see TListReportPage properties and methods

---

## EnumIndex Property (TListNode Object)

### Applies to

TListNode object

### Description

Returns the index of the node. It is a zero-based index enumerating all TList items disregarding their visibility (as if CurrentIndexMethod is set to TLSYS\_ENUM value). This property is read-only.

### Syntax

```
Index& = Node.EnumIndex  
Index& = Nodes(5).EnumIndex
```

### Data Type

Long

### Example

```
' The following provides the same result as printing Node.Text  
Index& = Node.EnumIndex  
Print TList.List(Index&)
```

---

## Environment Property

### Applies to

TList object

### Description

Due to differences in how different environments support OCX's, TList uses enumerated property **Environment**.

### Syntax

```
[form.]TList.Environment[=enum%]
```

### Settings

The **Environment** should be set to 0 or 1 depending on the development environment:

Setting	Description
0	(Default) VB, VC++, Delphi
1	FoxPro, this flag improves behaviour TList under FoxPro environment, prevents flickering when pressing PageUp/PageDown keys.
2	Internet Explorer, this flag improves behaviour TList on a Web page. PgUp and PgDown keys will work properly when the control is placed on a web page.

### Data Type

Integer

---

## Expand Event

### Applies to

TList object

### Description

This event is generated whenever an item is expanded, which means the item's subordinate items become visible.

### Syntax

```
Sub TList_Expand([Index As Integer,] ByVal J As Long)
```

### Remarks

This event passes *J*, the *index* of the item in the list that was expanded.

---

Note that TList's response to end-user click and double click events may or may not automatically expand or collapse the list depending on the setting of the **AutoExpand** property.

---



---

## Expand Property

### Applies to

TListNode object

### Description

Specifies whether an item is expanded (subordinate items visible). Setting the property to True will expand the outline (showing immediate children of the item) setting to False will collapse the outline at that point.

---

Note that upon expanding an item whose parent is collapsed, the following actions are taken:

- the parent is expanded;
- item is expanded.

---

Not available at design time, read/write at run-time.

### Syntax

```
[form.]TList.Expand(index&)[ = {True|False}]
[form.]TList.Nodes(index&).Expand[ = {True|False}]
```

### Settings

The **Expand** property settings are:

Setting	Description
---------	-------------

True	(Default)The item has expanded (visible) subordinate items.
False	The item's subordinate items, if any, are collapsed(hidden)

### Remarks

TList saves the collapsed/expanded state with an item when passing items via the clipboard, saving them to a file, or copying to or from a *tree buffer*.

To expand all subordinates (not just immediate children), use the **ExpandEx** property.

Setting of **Expand** property visibly updates the control unless the **Redraw** property is set to False.

---

Note that changing the **Expand** property may change the value of the **ListCount** property depending on the value of the **CurrentIndexMethod** property. For instance, with **CurrentIndexMethod** set to 1, setting **Expand** to True may increase the value of **ListCount** as children become visible.

---

This property is affected by the setting of the **ExpandChildren** property.

**Expand** property always returns False for invisible items.

### Example

This presumes AutoExpand is set to 0 or 2:

```
Sub TList_DblClick ()
    ' Expand or Collapse the tree in the Double Click event
    I% = TList.ListIndex ' identify the clicked item
    TList.Expand(I%) = Not TList.Expand(I%)
End Sub
```

### Data Type

Boolean

### See Also

**ExpandChildren** property

---

## ExpandChildren Property

### Applies to

TList object

### Description

This property allows TList to recall the expand/collapse state of a branch even after higher elements are collapsed.

If **ExpandChildren** = True, expanding a parent will return the expand/collapse state of the subordinate branches to their state before the parent was last collapsed.

### Syntax

```
[form.]TList.ExpandChildren [= bool%]
```

### Default

False

### Data Type

Boolean

---

## ExpandEx Property

### Applies to

TList object

### Description

Setting the **ExpandEx** property expands all subordinate elements of an item specified by its index. If the index is set to -1, it expands the entire list (as defined by the **CurrentIndexMethod** property and the **VisualRoot** property).

Not available at design time.

### Syntax

```
[form.]TList.ExpandEx(index&) [= bool%]
```

### Remarks

This property returns expand/collapse state of the item just as the **Expand** property does, but the **Expand** property always returns False for invisible items.

### Data Type

Boolean

---

## ExpandNewItem Property

### Applies to

TList object

### Description

The **ExpandNewItem** property determines the initial state of newly added items.

### Syntax

```
[form.]TList.ExpandNewItem [= bool%]
```

### Remarks

If **ExpandNewItem** is True then the following results in a fully expanded list:

```
Sub Form_Load()  
    TList.AddItem "Fred"  
    TList.AddItem "Fred's Daughter", 0  
    TList.AddItem "Fred's Grandson", 1  
End Sub
```

If **ExpandNewItem** is False then only Fred will be shown.

### Default

False

### Data Type

Boolean

---

## ExpandToLevel Property

### Applies to

TList object

### Description

The **ExpandToLevel** property expands and collapses all tree branches up to the specified level.

Note: if you set this property to 0 then TList collapses all the branches.

Not available at design time, read/write at run-time.

### Syntax

```
[form.]TList.ExpandToLevel [= Long&]
```

### Data Type

248 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX



Long

### See Also

**Expand**, **ExpandEx**, and **ExpandChildren** properties

---

## ExplorerCompatible Property

### Applies to

TList object

### Description

Makes TList look like Windows Explorer Outline.

### Syntax

[form.]TList.ExplorerCompatible [= enum%]

### Settings

The **ExplorerCompatible** property settings are:

Setting	Description
0 - None	(Default) Preserves TList's appearance as in earlier editions of TList.
1 - Keystrokes	TList processes all keystrokes which Window Explorer does.
2 - Tree Lines appearance	TList Tree Lines and their alignment look exactly as in Window Explorer.
3 - Keystrokes and Tree Lines appearance	Turns on both 1 and 2 options.

### Data Type

Integer

---

## FastAddItem and FastAddItemEx Methods

### Applies to

TList object

### Description

Obsolete see AddItem2 and AddItem2Ex functions description.

---

## File Property

### Applies to

TList object

### Description

The **File** property manages a TreePictureTable in which references to repeated images in a tree are held when saving the tree or *tree buffer* to a file. The purpose of the TreePictureTable is to optimize resource utilization when saving trees with many repeated images.

Write only at run-time, not available at design time.

### Syntax

[form.]TList.File(FileHandle%) = enum%

## Settings

The **File** property settings are:

Setting	Description
0	Load TreePictureTable.
1	Create TreePictureTable
2	Write TreePictureTable

## Remarks

Set the File property to 0 or 1 after opening a file from VB and before the first function call to read from or write to that file. Set the File property to 2 immediately before closing the file from VB.

## Example

```
Dim FreeHandle%
FreeHandle% = FreeFile
' Open file for output.
Open "e:\MyTListFile.tlt" For Binary Access Write As FreeHandle%
TList1.File(FreeHandle%) = 1 ' use 0 for input
TList1.Save(Index1) = FreeHandle%
TList1.Save(Index2) = FreeHandle%
TList2.Save(Index22) = FreeHandle%
TList2.Save(Index32) = FreeHandle%
TList1.File(FreeHandle%) = 2
Close FreeHandle% ' Close file.
```

## Data Type

Integer

---

# Files Property

## Applies To

**TListDataObject** object

## Description

Returns a collection of filenames used by the vbCFFiles (15) format which in turn contains a list of all filenames used by a TListDataObject; for example, the names of files that a user drags to or from the Windows File Explorer.

Not available at design time. Read-only

## Syntax

```
TListDataObject.Files(Index)
```

## Remarks

The **Files** collection is filled with filenames only when the **TListDataObject** contains data of type vbCFFiles. The **TListDataObject** object can contain several different types of data.

## Data Type

Object

---

# FindItem and FindValue Methods

## Applies to

**TList** object

## Description

These two functions may be used to search for a given item.

The **FindItem** method searches for an item based on the text (list property) of that item.

The **FindValue** method searches for an item based on its associated data (**ItemValues** property).

---

**Note:** FindItem method is limited to finding rows only in a single column list or tree, or in the first column (the Tree column ) of a grid. To find rows based on data in other columns FindValue method must be used.

---

### Syntax

```
[form.]TList.FindItem (ByVal FindWhat As String, ByVal nFlags As Integer,  
ByVal nFromIndex As Long, ByVal nToIndex As Long) As Long  
  
[form.]TList.FindValue (FindWhat As Variant, ByVal nFlags As Integer, ByVal nFromIndex As Long,  
ByVal nToIndex As Long, ByVal ValueName As Variant) As Long
```

### Parameters

Parameter	Description
<i>tlTree</i>	Name of the TList control.
<i>FindWhat</i>	String or Variant data being sought.
<i>nFlags</i>	How to search the item, This parameter is composed as the sum of bit flags - see description in the following table.
<i>nFromIndex</i>	Specifies the first item in the range.
<i>NToIndex</i>	Specifies the last item in the range.
<i>VaueName</i>	Optional. If present specifies the ValueName of the data to be searched.

The *nFlags* parameter flags:

Constant	Value	Description
TL_FI_DONTUSECASE	&H1	if set, search is not case-sensitive.
TL_FI_RELAXED	&H2	if set, valid item may include FindWhat as a substring of the item's text; otherwise the match must be exact.
TL_FI_STARTSWITH	&H4	if set TList searches items whose text starts with the <i>FindWhat</i> .
TL_FI_BACKDIR	H100	if set, TList searches backwards from the end of the range.
TL_FI_SELONLY	&H200	if set, TList searches only among selected items.

### Returns

Index of the first found item or -1 if item wasn't found.

---

## FireTListEvents Property (TListTreeView)

### Description

This property disables/enables firing TList events. TListTreeView events designed for Standard MS TreeView compatibility are unaffected by this property setting..

### Syntax

```
[form.]TListTreeView.FireTListEvents [= bool]
```

### Data Type

Boolean

---

## FirstItem and LastItem Properties

### Applies To

TListReport object

### Description

These properties specify the range of items to be printed.

Not available at design-time.

### Syntax

```
TListReportObject.FirstItem [ = ItemIndex&]  
TListReportObject.LastItem [ = ItemIndex&  
[form.]TList.Report.FirstItem [ = ItemIndex&]  
[form.]TList.Report.LastItem [ = ItemIndex&
```

### Data Type

Long

---

## FirstSibling and LastSibling Properties (TListNode Object)

### Applies to

TListNode object

### Description

Returns a reference to the corresponding (first or last) sibling of this node.

This property is read-only one.

### Syntax

```
Set TListNode = Node.FirstSibling  
Set TListNode = Node.LastSibling
```

### Remarks

Siblings are child items of a single parent item. All root items are siblings of one another. All immediate children of any specified item are siblings. The first sibling for an item is the first child item belonging to the same parent as the specified item. The last sibling is the last node of that parent.

The **FirstSibling**, and **LastSibling**, properties all return a reference to another **TListNode Object**.

### Data Type

TListNode

### See Also

Next, Prev and Parent properties

---

## FixedSize Property

### Applies to

TList object

### Description

This property determines whether all items in the control have the same height or not. When it is True, then all items have the same height. Otherwise, the height of the item is defined by the values

of **ItemImageDefHeight** and **ItemImageDefWidth** properties and by the real size of the item's picture, and height of the item's font.

Read only at run time.

### Syntax

```
[form.]TList.FixedSize[ = bool%]
```

### Data Type

Boolean

---

## FocusRectStyle property

### Applies to

TList object

### Description

**FocusRectStyle** property determines how focus rectangle will be drawn around items in the tree.

Note: for grid objects the focus rectangle is determined automatically accordingly to **ActivationMode** property, it will be drawn around the whole row area in case of row mode, around the cell area otherwise.

Prior to TList 8 this behavior was determined by the **InvStyle** property.

### Syntax

```
TList.FocusRectStyle [= enum%]
```

### Settings

**FocusRectStyle** property's settings are:

Constant	Setting	Description
TL_FOCUS_AROUND_ITEM_CELL	1	The focus rectangle is drawn around the text and picture (if exists) of the item.
TL_FOCUS_WHOLE_ITEM	0	(Default) The focus rectangle is drawn around the whole object area. Outside a Grid the object area is the entire width of the item. Within a Grid the object area may be a single cell or a complete row depending on the setting of the ActivationMode property.

### Data Type

Integer

---

## Font Property

### Applies To

TList Control

TListCellDef object

### Description

Returns a **Font** object.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Syntax

```
[form.]TList.Font  
[form.]TList.Grid.GridCellDef.Font  
[form.]TList.Grid.Cells(Row&,Col&).CellDef.Font
```

## Remarks

Use the **Font** property of an object to identify a specific **Font** object whose properties you want to use. For example, the following code changes the Bold property setting of a **Font** object identified by the **Font** property of a grid cell:

```
TList1.Grid.Cells(20,20).CellDef.Font.Bold = True
```

For more information, see the description of the **Font** property in the *Microsoft Visual Basic Language Reference*.

## See Also

**ItemFontName**, **ItemFontSize**, **FontName**, **FontSize**, **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, and **ItemFontUnder** properties

---

# FontBold, FontItalic, FontStrikethru, FontUnderline Properties

## Applies to

TList object

## Description

Determine default font styles in the following formats: **FontBold**, *FontItalic*, ~~FontStrikethru~~, and FontUnderline.

## Syntax

```
[form.]TList.FontBold [ = boolean%]  
[form.]TList.FontItalic [ = boolean%]  
[form.]TList.FontStrikethru [ = boolean%]  
[form.]TList.FontUnderline [ = boolean%]
```

## Settings

The properties settings are:

Setting	Description
True	Turns on the specified formatting style.
False	Turns off the specified formatting style.

## Remarks

Setting of these properties updates the control unless the **Redraw** property is set to False. For more information, see the description of the **FontBold**, **FontItalic**, **FontStrikeThru**, **FontUnderline** properties in the *Microsoft Visual Basic Language Reference*.

## Data Type

Boolean

## See Also

**ItemFontName**, **ItemFontSize**, **FontName**, **FontSize**, **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, and **ItemFontUnder** properties

---

## FontName Property

### Applies to

TList object

### Description

Determines the default font name.

### Syntax

```
[form.]TList.FontName [ = string_expression$]
```

### Remarks

Setting of **FontName** property updates the control unless the **Redraw** property is set to False. For more information, see the description of the **FontName** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

String

### See Also

**ItemFontSize**, **ItemFontName**, **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, **FontUnderline**, **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, and **ItemFontUnder** properties

---

## FontSize Property

### Applies to

TList object

### Description

Determines default font size.

### Syntax

```
[form.]TList.FontSize [ = numeric_expression%]
```

### Remarks

Setting of **FontSize** property updates the control, unless the **Redraw** property is set to False. For more information, see the description of the **FontName** property in the *Microsoft Visual Basic Language Reference*.

### Data Type

Integer

### See Also

**FontName**, **ItemFontName**, **ItemFontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, **FontUnderline**, **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, and **ItemFontUnder** properties

---

## Font3D property (TListCellDef object)

### Applies to

TListCellDef object

### Description

The **Font3D** property controls the presentation of either standard or 3-D text appearance in TList. Use of 3-D shadowed fonts can be helpful in display of text over a background image in TList.

**3-D** fonts may be set for the entire TList control or for any column, row, or specific cell (depending on the TList CellDef object this property applied to).

### Syntax

```
TListCellDefObject.Font3D = [enum]
```

### Settings

**Font3D** property's settings are:

Constant	Setting	Description
TL_FONT3D_NONE	0	(Default) Normal text.
TL_FONT3D_INSET	1	Inset text.
TL_FONT3D_RAISED	2	Raised Text.

### Example

```
'specify 3D appearance for all items in TList Tree or List
TList1.DefItemCellDef.Font3D = TL_FONT3D_RAISED

'specify 3D appearance for all items in TList Grid
tlist1.Grid.GridCellDef.Font3D = TL_FONT3D_INSET

'specify 3D appearance for a specific List or Tree item
TList1.ItemCell(1).Font3D = TL_FONT3D_INSET

'specify 2-D appearance for a specific Grid Cell
TList1.Grid.Cells(0,1).CellDef.Font3D = TL_FONT3D_NONE
```

### Data Type

Enumeration

### Remarks

**3-D** appearance is achieved using "shadows". A second presentation of the text is drawn offset both above and to left, or below and to right in a secondary shadow color depending on whether text is to appear raised or lowered. For non-standard shadow color use **FontShadowColor**, **FontShadowSelectedColor** properties. **FontShadowColor** and **FontShadowSelectedColor** determine the color of the shadow for 3D text for normal and selected items. Changing **Font3D** property alters **FontShadowColor**, **FontShadowSelectedColor** properties.

### See Also

*FontShadowColor, FontShadowSelectedColor properties.*

---

## FontShadowColor and FontShadowSelectedColor properties (TListCellDef object)

### Applies to

TListCellDef object

### Description

The **FontShadowColor** and **FontShadowSelectedColor** properties determine the shadow colors used to present 3D text for normal and selected items/cells/rows/columns.

This effect may be applied to the entire TList control or for any column, row, or specific cell (depending on the TList CellDef object this property applied to).



## Syntax

```
TListCellDefObject.FontShadowColor = [color]  
TListCellDefObject.FontShadowSelectedColor = [color]
```

## Example

```
'specify 3D appearance with non-standard colors for all List / Tree items ( also first column of Grid )  
TList1.DefItemCellDef.FontShadowColor = RGB(128, 192, 192)  
TList1.DefItemCellDef.FontShadowSelectedColor = RGB(192, 128, 128)
```

## Data Type

Color

## Remarks

**3-D** appearance is achieved using "shadows". A second presentation of the text is drawn offset both above and to left, or below and to right in a secondary shadow color depending on whether text is to appear raised or lowered. See also **Font3D** property.

---

# ForeColor Property

## Applies To

TList control

TListCellDef object

## Description

TLists's **ForeColor** property determines the default color in which text of an item is displayed.

TListCellDef **ForeColor** property determines the default color in which text of a cell is displayed.

## Syntax

```
[form.]TList.ForeColor[ = color&]  
[form.]TList.ItemCell(ItemIndex&).CellDef.ForeColor [ = color&]  
[form.]TList.Grid.GridCellDef.ForeColor [ = color&]  
[form.]TList.Grid.Cells(Row&, Col&).ForeColor [ = color&]  
[form.]TList.Grid.RowTitleCellDef.ForeColor [ = color&]  
[form.]TList.Grid.ColTitleCellDef.ForeColor [ = color&]
```

## Remarks

Setting of this property updates the control display unless the **Redraw** property is set to False.

For more information, see the description of the **ForeColor** property in the *Microsoft Visual Basic Language Reference*.

## Example

```
TList1.ForeColor = RGB(127, 0, 127)  
TList1.ForeColor = QBColor(2)
```

## Data Type

Long

## See Also

BackColor property

---

# Format Property

## Applies To

TListCellDef Object

## Description

The format property determines the visual text presented in a cell by applying a formatting string to the data content of a cell holding a numeric value.

---

TList's format property TList 8 now supports a much wider array of formats.

---

## Data Type

String

## Syntax

```
[form.]TList1.ItemCell( ItemIndex& ).Format [= str$]  
[form.]TList1.Grid.Cells(Row&, Col&).CellDef.Format [= str$]  
[form.]TList1.ItemGrid( ItemIndex& ).Cells(Row&, Col&).CellDef.Format [= str$]
```

## Settings

The format property is set with a string expression indicating how to display the contents of a cell holding a numeric value. The string expression may be made up of one to four elements, separated by semicolons. The element of the string is applied as a format is determined by whether the value of the cell is Positive, Negative, Zero or Null (not set).

For Example:

```
TList1.Grid.ColDefs(Column).CellDefs.format = _  
    "PlusFormat; MinusFormat; ZeroFormat; NullFormat"
```

In this case if the value of data in a cell in the specified column is positive the format string "PlusFormat" would be applied, if the value were negative "MinusFormat" would be applied, etc.

If one of the elements is not specified, the format specified by the first element is used. For example, in the following case the format string "UseMeForPlusOrMinus" will be used for formatting both Positive and Negative values since no 2<sup>nd</sup> element (for negative values handling) is provided within the format string:

```
TList1.Grid.ColDefs(Column).CellDefs.format = _  
    "UseMeForPlusOrMinus;; ZeroFormat; NullFormat"
```

In general each element of the format string expression may contain a combination of

- a) any desired static text – this is text that will appear exactly as entered in the Format string.
- b) and/or a reference to one of several predefined **Named Formats**  
(such as `"*Currency*"` or `"*Fixed*"` – see description and tables below),  
or a User-Defined Format string  
( such as `"*##,###.###"`, or `"*MMDDYYYY*"` – see description and tables below)

## For example:

```
FormatString = "static_text "  
FormatString = "*Named Format*"  
FormatString = "*UserDefinedFormatString*"  
FormatString = "static_text*Named Format*more_static_text"  
FormatString = "static_text*UserDefinedFormatString*more_static_text"
```

If no **Named Format** or **UserDefinedFormat** is specified within an element of the format string, only the static text from that element will be shown.

A **Named Format** refers to one of several standard predefined formats such as `"*Currency*"` or `"*Fixed*"` etc. (see table below). Such references must be surrounded by asterisks `"*"`. When a Named Format is applied by TList from within an element of a Format String, the numeric or date

typed data value actually assigned to the TList cell will be formatted (converted) according to the named format and surrounded by the static text for presentation.

For example, applying a format string with a first element (for positive values) of *"Price = \$\*Currency\* - a bargain"* to a TList cell with a value of 1000 would result in a display text of *"Price = \$1,000.00 - a bargain"*.

TList's **User-Defined** formats provide additional great flexibility in presenting numeric or date / time data. Using the user-defined format support you can easily display your own formats for numbers, dates, and times by specifying a custom format string between asterisks *"\*"*. You must use a special reserved symbols (explained in the tables below) to display parts of the Cell Value. The Format property converts the numeric or date-typed value to a text string and gives you control over the string's appearance. For example, you can specify the number of decimal places, leading or trailing zeros, and various date and time formats.

---

Note: The Number format symbols may be applied to Number data type values only, and Date/Time format symbols may be applied to Date/Time data type values only.

---

### Table Of Named Formats

The following table shows a number of named formats available to the user:

Data Type	Named Format	Description
Number	(Default) Empty string OR "Generic"	General format - Displays as entered.
Number	"*Currency*"	Display number with thousand separator, if appropriate; display two digits to the right of the decimal separator. Note that output is based on system locale settings.
Number	"*Fixed*"	Display at least one digit to the left and two digits to the right of the decimal separator.
Number	"*Standard*"	Display number with thousands separator, at least one digit to the left and two digits to the right of the decimal separator.
Number	"*Percent*"	Display number multiplied by 100 with a percent sign (%) appended to the right; always display two digits to the right of the decimal separator.
Number	"*Scientific*"	Use standard scientific notation.
Number	"*Yes/No*"	Display No if number is 0; otherwise, display Yes.
Number	"*True/False*"	Display False if number is 0; otherwise, display True.
Number	"*On/Off*"	Display Off if number is 0; otherwise, display On.
Date/Time	"*General Date*"	Display a date and/or time. For real numbers, display a date and time (for example, 4/3/93 05:34 PM); if there is no fractional part, display only a date (for example, 4/3/93); if there is no integer part, display time only (for example, 05:34 PM). Date display is determined by your system settings.
Date/Time	"Long Date*"	Display a date according to your system's long date format.
Date/Time	"*Medium Date*"	Display a date using the medium date format

		appropriate for the language version of Visual Basic.
Date/Time	"*Short Date"	Display a date using your system's short date format.
Date/Time	"*Long Time"	Display a time using your system's long time format: includes hours, minutes, seconds.
Date/Time	"*Short Time"	Display a time using the 24-hour format (for example, 17:45).

### **Table Of User-Defined Number Format Symbols**

The following table identifies characters you can use to create user-defined Number formats:

<b>Symbol</b>	<b>Description</b>
0	Digit placeholder; prints a trailing or a leading zero in this position, if appropriate.
#	Digit placeholder; never prints trailing or leading zeros.
.	Decimal placeholder.
,	Thousands separator.
- + \$ ( ) space	Literal character; characters are displayed exactly as typed into the format string.

### **Examples of user-defined Numeric Formats**

The following Table illustrate the result of applying Numeric Formats to the value 8315.4

<b>Format syntax</b>	<b>Result</b>
*00000.00*	08315.40
*#####.##*	8315.4
*##,##0.00*	8,315.40
*\$##0.00*	\$315.40

### **Table Of User-Defined Date / Time Format Symbols**

The following table identifies characters you can use to create user-defined date/time formats:

<b>Symbol</b>	<b>Description</b>
(:)	Time separator - In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.
(/)	Date separator - In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.
C	Display the date as dddd and display the time as tttt, in that order. Display only date information if there is no fractional part to the date serial number; display only time information if there is no integer portion.
D	Display the day as a number without a leading zero (1 – 31).
Dd	Display the day as a number with a leading zero (01 – 31).
Ddd	Display the day as an abbreviation (Sun – Sat).

Dddd	Display the day as a full name (Sunday – Saturday).
dddd	Display the date as a complete date (including day, month, and year), formatted according to your system's short date format setting. The default short date format is <code>m/d/yy</code> .
dddddd	Display a date serial number as a complete date (including day, month, and year) formatted according to the long date setting recognized by your system. The default long date format is <code>mmmm dd, yyyy</code> .
W	Display the day of the week as a number (1 for Sunday through 7 for Saturday).
Ww	Display the week of the year as a number (1 – 54).
M	Display the month as a number without a leading zero (1 – 12). If <code>m</code> immediately follows <code>h</code> or <code>hh</code> , the minute rather than the month is displayed.
Mm	Display the month as a number with a leading zero (01 – 12). If <code>m</code> immediately follows <code>h</code> or <code>hh</code> , the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan – Dec).
mmmm	Display the month as a full month name (January – December).
q	Display the quarter of the year as a number (1 – 4).
y	Display the day of the year as a number (1 – 366).
yy	Display the year as a 2-digit number (00 – 99).
yyyy	Display the year as a 4-digit number (100 – 9999).
h	Display the hour as a number without leading zeros (0 – 23).
hh	Display the hour as a number with leading zeros (00 – 23).
n	Display the minute as a number without leading zeros (0 – 59).
nn	Display the minute as a number with leading zeros (00 – 59).
s	Display the second as a number without leading zeros (0 – 59).
ss	Display the second as a number with leading zeros (00 – 59).
tttt	Display a time as a complete time (including hour, minute, and second), formatted using the time separator defined by the time format recognized by your system. A leading zero is displayed if the leading zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is <code>h:mm:ss</code> .
AM/PM	Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M.
am/pm	Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M.
A/P	Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M.
a/p	Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 P.M.
AMPM	Use the 12-hour clock and display the AM string literal as defined by your system with any hour before noon; display the PM string literal as defined by your system with any hour between noon and 11:59

P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. The default format is AM/PM.

### Examples of user-defined Date/Time Formats

The following Table illustrate the result of applying Date/Time Formats to the value, Wednesday, January 27, 1993

Format syntax	Result
*m/d/yy*	1/27/93
*dddd, mmmm dd, yyyy*	Wednesday, January 27, 1993
*d-mmm*	27-Jan
*mmmm-yy*	January-93
*hh:mm AM/PM*	07:18 AM
*h:mm:ss a/p*	7:18:00 a
*d-mmmm h:mm*	3-January 7:18

### Remarks

The asterisk,"\*", is a special character. To have TList present an asterisk in the displayed text use the string "/" as part of the format string.

\*\*\*\* TList cells containing non-numeric values will **not** be affected by the **Format** property. Thus data set with the List property or AddItem method of TList is **not** affected by the **Format** property of TList.

### Example

```
' The following code resets a Format String:
TList1.Grid.GridCellDef.Format = ""

' Format Column 1 of Grid to display a Short date
TList1.Grid.ColDefs(1).CellDef.Format = "*Short Date*"

' Format entire grid to show text "Empty" in all empty cells:
TList1.Grid.GridCellDef.Format = ";;;Empty"

' Display "Not Null" for all positive values,
' Display Negative and Zero values in Generic format
' and Display "Null for Empty Cells
Format = "NotNull;;;Null"

' Display "Plus Format" for all Positive values"
' "Minus Format" for all Negative Values
' "Zero Format" for both Zero and for Null values
Format = "PlusFormat; MinusFormat; ZeroFormat; ZeroFormat"

' Below is a sample how you can emulate "Yep/Nope" format:
' This code will show "Yep" in cells containing positive or negative values,
' and Nope for cells containing a value of zero, or no value at all (null).
TList1.Grid.ColDefs(1).CellDef.Format = "Yep;Yep;Nope;Nope"
```

---

## Format Property (TListDateTime Object)

### Applies to

TListDateTime object

## Description

This property specifies the format of a string representing the date in the editing window.

## Values

The **Format** property settings are:

Constant	Value	Description
TLFORMAT_USE_CELLDEF_FORMAT	-1	(Default value) If this flag is set, the format in the editing window corresponds to the display format of the unedited cell (as specified by the CellDef.Format property) If the CellDef.format is not set or is not a valid Date/Time format, then the editing format is taken from the DateTime. <b>Format</b> property. If the <b>Format</b> property is not set, the "General Date" format is used.
TLFORMAT_GENERAL_DATE	0	Displays date and/or time. For real numbers, displays a date and time (for example, 4/3/93 05:34 PM); if there is no fractional part, displays date only (for example, 4/3/93); if there is no integer part, time only is displayed (for example, 05:34 PM). The date format is defined by system settings
TLFORMAT_LONG_DATE	1	Displays date according to system long date format.
TLFORMAT_MEDIUM_DATE	2	Displays date using the medium date format appropriate for the language version of Visual Basic
TLFORMAT_SHORT_DATE	3	Displays date using system short date format.
TLFORMAT_LONG_TIME	4	Displays time using system long time format: including hours, minutes, seconds.
TLFORMAT_MEDIUM_TIME	5	Displays time using system time format without seconds (for example, 5:45 PM).
TLFORMAT_SHORT_TIME	6	Displays time using the 24-hour format (for example, 17:45).
TLFORMAT_CUSTOM	7	Applies the format specified by the the <b>FormatString</b> property.

## Syntax

[TListCellDef].EditInfo.DateTime. <b>Format</b> [= enum%]
---

## Data Type

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

Integer

## REMARKS

The Format property only controls the display of the data during editing. Once editing is completed the display is formatted according to the Format of the cell itself rather than of the edit object.

The Format property of the DateTime object defaults to the format associated with the CellDef object (TLFORMAT\_USE\_CELLDEF\_FORMAT value) - the format used for general display of the cell). If the CellDef.Format is not set, or is not set to a date/time type format, or if the cell value can not be correctly transformed to the Date/Time data type then the TLFORMAT\_GENERAL\_DATE format will be used for display during editing. Changing the Format property of the DateTime object overrides the Format property of the CellDef object.

When the initial cell value ( prior to editing) can not be correctly transformed to the Date/Time data type, the initial value for editing will be the current system time.

---

## FormatString Property (TListDateTime Object)

### Applies to

TListDateTime object

### Description

This property specifies a custom format for the string in the editing window. The property takes effect only when the **Format** property is set to TLFORMAT\_CUSTOM.

### Values

Day, Month, Year and Era symbolic notations

Notation	Description
Day settings	
d	Date of the month as digits without leading zeros for single digit days.
dd	Date of the month as digits with leading zeros for single digit days
ddd	Day of the week as a 3-letter abbreviation
dddd	Day of the week as complete word
<b>Month settings</b>	
M	Month as digits without leading zeros for single digit months.
MM	Month as digits with leading zeros for single digit months
MMM	Month as a three letter abbreviation.
MMMM	Month as complete word.
<b>Year settings</b>	
y	Year represented only be the last digit.
yy	Year represented only be the last two digits.
yyyy	Year represented by the full 4 digits.



## Hour, Minute and Second symbolic notations

Notation	Description
Hours settings	
h	Hours without leading zeros for single-digit hours (12-hour clock).
hh	Hours with leading zeros for single-digit hours (12-hour clock).
H	Hours without leading zeros for single-digit hours (24-hour clock).
HH	Hours with leading zeros for single-digit hours (24-hour clock).
Minutes settings	
m	Minutes without leading zeros for single-digit minutes.
Mm	Minutes with leading zeros for single-digit minutes.
Seconds settings	
s	Seconds without leading zeros for single-digit seconds.
ss	Seconds with leading zeros for single-digit seconds.
Time Marker settings	
t	One character-time marker string (that is, AM is displayed as "A").
tt	Multicharacter-time marker string (that is, AM is displayed as "AM").

## Syntax

```
[TListCellDef].EditInfo.DateTime.FormatString [= String$]
```

## Data Type

String

## Example

As a result of setting the FormatString as below, a user editing the cell will see a display in the edit area of "25th day of year 2001, 12 hours and 22 minutes."

```
With TList1.ItemCell(3).EditInfo
    .Style = TLEDITINFO_DATE_TIME
    .DateTime.Format = TLFORMAT_CUSTOM
    .DateTime.FormatString = "dd'th day of year 'yyyy'," _
        & " 'H' hours and 'mm' minutes"
End With
```

## REMARKS

If the **FormatString** property is not set but the **Format** property is set to TLFORMAT\_CUSTOM, data will be represented in the "General Date" format.

The Format and FormatString properties only control the display of the data during editing. Once editing is completed the display is formatted according to the Format property of the cell itself rather than of the edit object.

---

## FreeBuffer Method

### Applies to

TList object

### Description

This method frees the *tree buffer* and the memory associated with it.  
Not available at design time.

### Syntax

```
[form.]TList1.FreeBuffer(ByVal hTreeBuffer As Long)
```

### Remarks

This method should be called for each *tree buffer* after the *tree buffer* is no longer needed.  
The parameter *htree buffer* is a variable of a *Long* type.

---

You can use any TList control to free any *tree buffer*.

---

---

## FreeBuffer/TListFreeBuffer improvement

### Applies to

TList object

### Description

A special value of **-1** is now supported for both the TListFreeBuffer function and the FreeBuffer method frees all buffers belonging to all TList controls of the current application.  
See also FreeBuffer method description.

### Example

```
Call TList1.FreeBuffer(-1)
Or
Call TListFreeBuffer(-1)
```

---

## FullItemString and FullRowString Properties

### Applies to

TListGrid object

### Description

These properties return a delimited string containing the data from each column of a row concatenated using the **ColDelimiter** character to separate column values. Thus they perform the exact reverse procedure of how TList parses delimited data when using **AddItem** or **AddRow** methods.

### Syntax

```
TList.FullItemString(ByVal Index As Long) As String
TListGrid.FullRowString(ByValRowIndex As Long) As String
```

### Remarks

If an ItemGrid exists within a TreeGrid, a single item may have multiple ItemGrid columns within the first column of the TreeGrid. In this case **FillItemString** only returns the column data for the TreeGrid – this includes the first column of the ItemGrid but excludes additional ItemGrid columns.

---

## FullPath Property

### Description

Returns the *fully qualified* name of an item. The fully qualified name is the concatenation of the item with its parent item, the parent item's parent item, and so on until the parent item at indentation level 0 is reached. The **FullPath** property is an array whose *index* values correspond to the items in the list.

Not available at design time and read-only at run time.

### Syntax

<code>[form.]TList.FullPath(index&amp;)</code>
--

### Remarks

Use the **PathSeparator** property to create a delimiter between the components of the **FullPath** property. This is useful when the TList control contains file-system components such as directory names and file names.

Note the **FullPath** property doesn't depend on the **VisualRoot** property settings.

### Data Type

String

---

## GetArrayProperty, SetArrayProperty, GetArrayPropertyID Properties

### Applies to

TList object

### Description

These three properties provide you with an access to the TList array properties by property name. This feature was specifically added for FoxPro users although they can also be used within other programming environments. For some internal reasons aren't connected with TList control itself FoxPro doesn't allow users to work with some array properties with an index greater 65000. In order to solve these problems TList provides GetArrayProperty, SetArrayProperty, GetArrayPropertyID properties that allow you to work with a control under FoxPro environment without index range limitations.

**GetArrayProperty** gets the value to the property specified by the name or ID (returned via GetArrayPropertyID call).

**SetArrayProperty** sets the value to the property specified by the name or ID (returned via GetArrayPropertyID call).

**GetArrayPropertyID** property allows you to get an ID of property in order to speed up accessing these properties for multiple calls (GetArrayProperty/SetArrayProperty properties work faster when you specify ID instead of the name of the property).

Note: The name of the property should be spelled exactly as it is written in document or shown in Object Browser. In case of using wrong name a corresponding error will be generated. Also it is the programmer's responsibility to pass the correct type of the variable(data) in order to get/set a

property. Please consult with TList documentation in order to get to know the correct type of the data that can be set to or gotten from a particular property.

### Syntax

```
TList1.GetArrayProperty(ByVal varPropertyNameOrID As Variant, ByVal IPropertyIndex As Long) As Variant  
TList1.SetArrayProperty(ByVal varPropertyNameOrID As Variant, ByVal IPropertyIndex As Long) As Variant  
TList1.GetArrayPropertyID(ByVal varPropertyName As String) As Variant
```

### Example

Change the List property for items in the range 50000 to 100000

1) VB sample

```
Dim idList as Variant  
idList = TList1.GetArrayPropertyID("List")  
Dim iCnt as Long  
For iCnt = 50000 To 100000  
    TList1.SetArrayProperty(idList, iCnt, "Item" & iCnt)  
Next
```

2) FoxPro sample

```
LOCAL idList  
LOCAL iCnt  
idList = thisform.tlist.GetArrayPropertyID([List])  
FOR iCnt = 50000 TO 100000  
    thisform.tlist.SetArrayProperty(idList, iCnt, [Item] + str(iCnt))  
ENDFOR
```

---

## GetData Method

### Applies To

TListDataObject Object

### Description

Returns data from a TListDataObject object.

### Syntax

```
TListDataObject.GetData (ByVal Format As Variant)
```

### Remarks

The **GetData** method syntax has these parts:

Part	Description
<i>Format</i>	A constant or value that specifies the Clipboard data format, as described in Settings.

### Settings

The settings for the *Format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (.dib files)
vbCFFiles	15	Filenames of dropped files. They can be accessed via Files property.
vbCFText	1	ASCII Text.

---

# GetFormat Method

## Applies To

TListDataObject Object

## Description

The method determines whether a TListDataObject object has data in a specified format.

## Syntax

<i>TListDataObject</i> .GetFormat (ByVal Format As Variant)
---

## Remarks

The **GetFormat** method syntax has these parts:

Part	Description
<i>Format</i>	A constant or value that specifies the Clipboard data format, as described in Settings.

## Settings

The settings for the *Format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (.dib files)
vbCFFiles	15	Filenames of dropped files. They can be accessed via <b>Files</b> property.
vbCFText	1	ASCII Text.

## Data Type

Boolean

---

# GetItemByCellValue Method (TListComboItems)

## Applies to

TListComboItems object

## Description

This method returns the reference to the **TListComboItem** object with the given unique key (**CellValue**).

## Syntax

<code>[TListComboItem] = [TListEditInfo].EditInfo.ComboBox.Items.<b>GetItemByCellValue</b>(CellValue)</code> where <b>CellValue</b> is the value corresponding to the value of the <b>TListComboItem</b> object. <b>CellValue</b> and to the first parameter of the <b>Add</b> method of the <b>TListComboItem</b> object..
--

## Example

<pre>'add four items to the combo list with pictures [TListEditInfo].ComboBox.Items.Add 10, Picture1.Picture, "Ten" [TListEditInfo].ComboBox.Items.Add 20, Picture2.Picture, "Twenty" [TListEditInfo].ComboBox.Items.Add 30, Picture3.Picture, "Thirty" [TListEditInfo].ComboBox.Items.Add 40, Picture4.Picture, "Forty" '... some other code ' * Returns the item "Twenty" from the combo list Dim objItem as TListComboItem</pre>
---

```
objItem = [TListEditInfo].ComboBox.Items.GetItemByCellValue(20)
```

```
...
```

## GetItemByXY Method

### Applies to

TList object

### Description

The **GetItemByXY** method returns the index of an item given a set of X/Y coordinates in pixels. It can also be used during mouse events to determine whether an X/Y coordinate is over a picture. Not available at design time.

### Syntax

```
[form.]TList.GetItemByXY(ByVal X As Integer, ByVal Y As Integer, ByVal nType As Integer) As Long
```

### Declarations

nType parameter constants:

Constant	Value	Description
PTFIND_ITEM	1	Point is within item's area
PTFIND_PIC	2	Point is within item's picture area
PTFIND_TEXT	4	Point is within item's text area
PTFIND_ITEM_VIS_ENTIRELY Y	8	Point is within item's area and item is entirely visible

### Parameters

X, Y are the control's internal coordinates in pixels. The 0, 0 point is at the left, top corner of the control.

### Returns

If point (X, Y) is within item's area and nType = PTFIND\_ITEM, return value = index of item;  
If point (X, Y) is within item's picture area and nType = PTFIND\_PIC, return value = index of item.  
If point (X, Y) is within item's text area and nType = PTFIND\_TEXT, return value = index of item.  
Otherwise, If no item is under specified coordinates the function returns -1.

### Example

```
Sub TList1_MouseMove (Button As Integer, _  
    Shift As Integer, X As Single, Y As Single)  
    ' Demonstrate GetItemByXY method  
    nX = X / Screen.TwipsPerPixelX  
    nY = Y / Screen.TwipsPerPixelY  
    nType = PTFind_Item 'or set to PTFind_Pic for picture only.  
    Item_Index = TList1.GetItemByXY(nX, nY, nType)  
    ' the alternative variant is  
    ' Item_Index = TList1.GetItemByXY(nX, nY, nType)  
    Text1.Text = "The mouse is over index number: " _  
        & Str$(Item_Index)  
End Sub  
  
Sub TList1_MouseDown (Button As Integer, _  
    Shift As Integer, X As Single, Y As Single)  
    nX = X / Screen.TwipsPerPixelX  
    nY = Y / Screen.TwipsPerPixelY  
    nType = PTFind_Pic 'Find item for picture only.  
    Index& = TList1.GetItemByXY(nX, nY, nType)
```

```
' the alternative variant is
' Item_Index& = TList1.GetItemByXY(nX, nY, nType)
Text1.Text = "MouseDown over picture: " & Str$(Index&)
End Sub
```

## GetItemRect Method

### Applies to

**TList** object

### Description

The **GetItemRect** method returns (as a parameter) a structure describing the rectangular region containing a specific item in the outline. The method returns 0 if the call was successful and an error code otherwise. All coordinates are measured in pixels.

Not available at design time.

### Declarations

```
[form.]TList.GetItemRect(
    ByVal nIndex As Long, ByVal nOpts As Integer,
    pRect As RECT) As Integer
```

### Parameters

*tlTree* - TList control.

*nIndex* - index of the item.

*nOpts* - one of the following options:

Constant	Value	Description
TLGR_ITEM	&H1	The size & coordinates of the whole item will be returned.
TLGR_PLUSMINUS	&H2	The size & coordinates of the plus-minus picture of the item will be returned.
TLGR_MARK	&H4	The size & coordinates of the mark picture of the item will be returned.
TLGR_PICTURE	&H8	The size & coordinates of the picture of the item will be returned.
TLGR_TEXT	&H10	The size & coordinates of the text of the item will be returned.

In OCX version of TList this function was replaced with Visual Basic function with the same name. This function can be found in the TLIST8.BAS file.

Note that in the OCX edition, pRect.Left should be passed as a parameter instead of pRect. In this case pRect.Left acts as a pointer to the structure as a whole.

## GotFocus Event

### Applies to

**TList** object

### Description

Occurs when the TList control receives the focus.

### Syntax

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

```
Sub TList_GotFocus([Index As Integer])
```

## Remarks

For more information, see the description of the **GotFocus** event in the *Microsoft Visual Basic Language Reference*.

# GradientColorFrom, GradientColorTo, and GradientStyle Properties

## Applies To

TList Control

TListCellDef object

## Description

These properties if applied to TList Control allow painting gradually changing colors on the TList background.

These properties if applied to **TListCellDef** allow painting gradually changing colors on this cell background. Thus you can specify different gradient colors and the style for any cell belonging the grid or tree item.

## Syntax

```
[form.]TList.GradientColorFrom [ = color&]  
[form.]TList.GradientColorTo [ = color&]  
[form.]TList.GradientStyle [ = enum%]  
  
[form.]TList1.Grid.Cells(R, C).CellDef.GradientColorFrom = [color&]  
[form.]TList1.Grid.Cells(R, C).CellDef.GradientColorTo = [color&]  
[form.]TList1.Grid.Cells(R, C).CellDef.GradientStyle = [enum%]  
  
[form.]TList1.Grid.ColDef(C).CellDef.GradientColorFrom = [color&]  
[form.]TList1.Grid.ColDef(C).CellDef.GradientColorTo = [color&]  
[form.]TList1.Grid.ColDef(C).CellDef.GradientStyle = [enum%]  
  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs(C).CellDef.GradientColorFrom = [color&]  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs(C).CellDef.GradientColorTo = [color&]  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs(C).CellDef.GradientStyle = [enum%]  
  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.GradientColorFrom = [color&]  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.GradientColorTo = [color&]  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.GradientStyle = [enum%]  
  
[form.]TList1.ItemCellDef(ItemIndex&).GradientColorFrom = [color&]  
[form.]TList1.ItemCellDef(ItemIndex&).GradientColorTo = [color&]  
[form.]TList1.ItemCellDef(ItemIndex&).GradientStyle = [enum%]  
  
[form.]TList1.DefItemCellDef.GradientColorFrom = [color&]  
[form.]TList1.DefItemCellDef.GradientColorTo = [color&]  
[form.]TList1.DefItemCellDef.GradientStyle = [enum%]
```

## Settings

The **GradientStyle** property settings are:

Setting	Value	Description
TLGRADIENT_NONE	0	Default. Doesn't paint gradient background.
TLGRADIENT_LEFT_TO_RIGHT	1	Colors are gradually changing from the left to right.



TLGRADIENT_TOP_TO_BOTTOM	2	Colors are gradually changing from the top to the bottom.
TLGRADIENT_CENTER_HORZ	3	Colors are gradually changing from the center in horizontal direction.
TLGRADIENT_CENTER_VERT	4	Colors are gradually changing from the center in vertical direction.

### Data Type

Long

---

## Grid Property

### Applies To

TList control

TListGridCell object

TListColDef object

TListNode object

### Description

Returns a reference to a Grid object.

The Grid property may also be set to the Visual Basic constant, **Nothing**, thereby removing Grid formatting. Otherwise it is a read-only property:

### Syntax

```
[form.]TList1.Grid
[form.]TList1.ItemGrid(ItemIndex&)
```

### Examples

```
TList1.Grid.Cols = 3 ' This will create a grid with 3 cols
TList1.Grid = Nothing ' This will destroy the grid

Dim x as TListGrid ' Dim X as a TListGrid object
Set X = TList1.ItemGrid(3) ' this returns reference to an itemgrid
```

### Data Type

Object

---

## GridCellActivate Event / GridCellDeactivate Event

### Applies to

TListGrid object

### Description

The **GridCellActivate** event occurs right after a cell becomes active (having focus), but not necessary selected for multi-selection modes) but before any changes are displayed on the screen. The **GridCellDeactivate** event when a cell is being deactivated ( losing focus), but before the corresponding changes become visible on the screen.

### Syntax

```
Sub TList_GridCellActivate([Index As Integer], ByVal objGridCell As TListGridCell, ByVal IReserved As Long)
Sub TList_GridCellDeactivate([Index As Integer], ByVal objGridCell As TListGridCell, ByVal IReserved As Long)
```

### Parameters

Parameter	Description
-----------	-------------

objGridCell	Returns a reference to the cell that is activated / deactivated.
IReserved	Reserved for this version.

### Remarks

The **GridCellActivate** event won't be triggered if the cell or set of cells is not enabled for activating (see **Activatable** property for details). Use the **GridCellClick** event for trapping clicks over disabled cells.

The **ActiveCell**, or **Row** and **Col** properties are updated after the **GridCellDeactivate** event, but before the **GridCellActivate** event. Thus the previously Active Cell can still be referenced via these properties inside the GridCellDeactivate event.

---

*Note:* Upon entry to this event *objGridCell* parameter will be initially set by TList to *Nothing* if the event was triggered as a result of deleting the corresponding GridCell object.

---

### Example

```
' Select / Deselect Grid Cell upon Activation ( as user Navigates for instance)
' Normally TList may just show a focus rectangle

Sub TList1_GridCellActivate( ByVal objGridCell As TListProLibCtl.TListGridCell, ByVal IReserved As Long)
    objGridCell.Selected = True
End Sub

Sub TList1_GridCellDeactivate( ByVal objGridCell As TListProLibCtl.TListGridCell, ByVal IReserved As Long)
    If Not objGridCell Is Nothing Then
        objGridCell.Selected = False
    End If
End Sub
```

## GridCellClick Event

### Applies to

TListGrid object

### Description

This event occurs when the user selects a cell ( except a Row or Column Title) in any Grid of a TList control by clicking the mouse button.

---

**Syntax change** – this event is no longer triggered when user clicks on a Row Title or Column Title Cell. Instead the **GridColumnTitleClick**, **GridRowTitleClick** or **GridCornerTitleClick** event is triggered.

---

### Syntax

```
Sub TList_GridCellClick([Index As Integer], ByVal objGridCell As TListGridCell, ByVal Button As Integer)
```

### Parameters

Parameter	Description
objGridCell	Returns a reference to the <b>TListGridCell</b> object for the clicked cell.
Button	returns an integer identifying which button was pressed. This value is composed of the following bit fields: left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively.

### Remarks

**Important:** The `TList.ActiveGrid`, `Grid.ActiveRow`, `Grid.Col` and `Grid.Row` properties should not be used within this event – such use may result in unpredicted ( and unsupported) behavior. This event passes *GridCell*, the reference to a `TListGridCell` object of the cell, which was clicked. The `.Row` and `.Col` properties of the *GridCell* object can be used to determine which cell was clicked. You can find out what Grid object this cell belongs to using **GridCell.Grid** property. Clicking over the cell may not change activation or selection for this cell; both of these actions depend on the values of **Selectable**, **Activatable** and **ActivationMode** properties. If the cell cannot be activated the `GridCellActivate` event will not be triggered when clicking on the cell. Note neither the `GridCellClick` nor the `GridCellDbClick`, events are fired for cells within the Column Title or Row Titles. For such cells trap the `GridColumnTitleClick`, `GridRowTitleClick` or `GridCornerTitleClick` events.

### Example

```
Sub TList_GridCellClick, ByVal objGridCell As TListGridCell, ByVal Button As Integer)
    RowNumber = ObjGridCell.Row
    ItemIndex = objGridCell.Grid.RowToItemIndex(RowNumber)
    ColumnNumber = ObjGridCell.Col
    ColumnName = objGridCell.Grid.ColDefs(ColumnNumber).ValueName
End Sub
```

### See Also

`GridRowTitleClick` Event, `GridColumnTitleClick` Event, `GridCornerTitleClick` Event

## GridCellDbClick Event

### Applies to

`TListGrid` object

### Description

This event occurs when the user double clicks a cell in any Grid of `TList` control, either by pressing the arrow keys or by clicking the mouse button.

### Syntax

```
Sub TList_GridCellDbClick([Index As Integer], ByVal GridCell As TListGridCell,
    ➡ ByVal Button As Integer)
```

### Remarks

This event passes *GridCell*, the reference to a `TListGridCell` object of the cell, which was double clicked. You can find out what Grid object this cell belongs to using **GridCell.Grid** property. The *Button* parameter returns an integer identifying which button was pressed. This value is composed of the following bit fields: left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Note neither the `GridCellClick` nor the `GridCellDbClick`, events are fired for cells within the Column Title or Row Titles. For such cells trap the `GridColumnTitleClick`, `GridRowTitleClick` or `GridCornerTitleClick` events.

## GridCellDef Property

### Applies To

`TListGrid` object

### Description

This property returns a reference to a **TListCellDef** object. This object is used to collect all properties which determine how a grid cells will be displayed by default. This property doesn't affect row titles or column titles defaults.

### Syntax

```
[form.]TList1.Grid.GridCellDef  
[form.]TList1.ItemGrid(ItemIndex&).GridCellDef
```

### Example

```
TList1.Grid.GridCellDef.BackColor = RGB(127, 127, 127)
```

## GridCellRequestEditing Event

### Applies to

TListGridCell object

### Description

This event is triggered when the *TListGridObject.CellEdit* property is set to TL\_CELLEEDIT\_BEGIN but before any actual editing takes place.

### Syntax

```
Sub TList_GridCellRequestEditing (Cancel As Integer,  
    vTextToEdit As Variant, ByVal objGridCell As TListGridCell,  
    vOptions As Variant)
```

### Parameters

Parameter	Description
Cancel	Initially set to <u>False</u> . If you set it to <u>True</u> , editing is canceled.
vTextToEdit	Initially this is the text of the cell being edited, but you can modify this text, changing what is seen in the edit box.
objGridCell	TListGridCell object representing the cell being edited.
vOption	This parameter is a sum composed of specified flag bits (see following table), you can change any of these settings in the GridCellRequestEditing event procedure. <b>Note:</b> some of these flags can be ignored. This limitations depend on the type of the editor is called for editing the cell (no limitations only for textbox editors). See TListEditInfo object for details.

*vOption* flags:

Constant	Value	Description
TL_REQED_LEFT	&H0	Left aligns text.
TL_REQED_CENTER	&H1	Centers text.
TL_REQED_RIGHT	&H2	Right aligns text.
TL_REQED_MULTILINE	&H4	Designates multiline editing.
TL_REQED_UPPERCASE	&H8	Converts all characters to uppercase as they are typed.
TL_REQED_LOWERCASE	&H10	Converts all characters to lowercase as they are typed.
TL_REQED_PASSWORD	&H20	Displays all characters as an asterisk (*) as they are typed.
TL_REQED_AUTOVSCROLL	&H40	If this flag is specified, the control automatically scrolls horizontally when

		the caret goes past the right edge of the control. To start a new line, the user must press ENTER.
TL_REQED_AUTOHSCROLL	&H80	If this flag is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses ENTER. The position of the word-wrap is determined by the item size.
TL_REQED_READONLY	&H800	Prevents the user from typing or editing text.
TL_REQED_STDCOLORS	&H100	Displays edited item using standard colors.
TL_REQED_BORDER	&H200	Draws border around edited item.
TL_REQED_VSCROLLER	&H400	Displays vertical scrollbar while item is edited.
TL_REQED_HSCROLLER	&H200 0	Displays horizontal scrollbar while item is edited.
TL_REQED_NOMENU	&H400 0	Disable showing right click menu during editing operation.
TL_REQED_FULLRECT	&H800 0	If the flag is specified, size of edit window equals to size the grid cell.

### See Also

GridCellAfterEditing, GridCellEditingKeyDown, GridCellEditingKeyUp and GridCellEditingKeyPress events, EditingMode and CellEdit properties

---

## GridCellAfterEditing Event

### Applies to

TListGridCell object

### Description

The **GridCellAfterEditing** event is triggered when the *TListGridObject.CellEdit* property is set to TLEDITMODE\_END or when editing is canceled by the user by clicking on another control, scrolling the control, etc.

### Syntax

```
Sub TList1_GridCellAfterEditing (ByVal vEditedText As Variant,vConvertedText As Variant,  
ByVal objGridCell As TListGridCell,ByVal CancelledBy As Integer)
```

### Parameters

Parameter	Description
vEditedText	Contains the string representation of data as just entered by the end-user.
vConvertedText	Contains the edited text converted by TList accordingly to the cell data type. This data will be placed into the corresponding grid cell. This may differ from <i>EditedText</i> due to Formatting.
objGridCell	A TListGridCell object representing the cell being edited.
CancelledBy	Specifies how editing operation was concluded - Returns 0 if editing was canceled programmatically by setting one of the properties, -Returns 1 if the editing was ended by user.

---

**Note 1:** If it is impossible to convert the edited text accordingly to the grid cell data type, `vConvertedText` parameter will contain the same data as `vEditText` parameter and user have to modify this data in order to solve this situation.

**Note 2:** It is possible to force resumption of editing within the `GridCellAfterEditing` Event (for example, after a wrong value is entered by an end user), by setting the `CellEdit` property to `TLEDITMODE_CONTINUE` (= 3). You can display a message box from within this event in order to notify users.

**Note 3:** While editing checkboxes `AfterEditing` event will be fired with `vEditText` parameter that contains a checkbox value but a checkbox visible caption. This value also can be addressed by **CheckboxValue** property and can be set to 0,1 or 2 for standard checkboxes. For using non-standard values for checkboxes see **UncheckedValue**, **CheckedValue** and **GrayedValue** properties of **TListCheckbox** object.

---

### Example

```
Private Sub TList1_GridCellAfterEditing(ByVal vTextToEdit As Variant, vConvertedText As Variant, _
    ByVal objGridCell As TListProLibCtl.TListGridCell, ByVal CancelledBy As Integer)
    'new text can't be shorter than 5 characters
    If Len(vConvertedText) < 5 Then
        MsgBox "The text size should be at least 5 characters"
        TList1.Grid.dit(AnyIndex, AnyIndex) = TLEDITMODE_CONTINUE 'continue editing of this item
    End If
End Sub
```

---

**Note:** note that the **Row** and **Col** parameters of the `CellEdit` property are ignored when setting this property to `TLEDITMODE_CONTINUE`

---

### Remarks

Setting the **CellEdit** to `TLEDITMODE_CANCEL` does not trigger this event.

### See Also

`GridCellRequestEditing`, `GridCellEditingKeyDown`, `GridCellEditingKeyUp` and `GridCellEditingKeyPress` events,  
`EditingMode` and `CellEdit` properties

---

## GridCellEditingChange Event

### Applies to

**TListGridCell** object

### Description

This event is triggered in response to end-user editing changes (via keyboard or mouse) that occurs while the user is editing grid cells using a built-in editor (**TListTextBox**, **TListComboBox**, **TListCheckBox**, etc).

The event may be used to validate the data being entered by end-user and to correct the data if necessary during editing.

### Syntax

```
Sub TList_GridCellEditingChange (ByVal ItemIndex As Long,
    ➔ ByVal objGridCell As TListGridCell, ByVal objChangeInfo As ➔ TListEditingChangeInfo)
```

### Parameters

Parameter	Description
ItemIndex	Index of the item that corresponds to the row of the grid cell that is being edited.
GridCell	<b>TListGridCell</b> object referencing the cell being edited.

ObjChangeInfo

**A TListEditingChangeInfo** object providing access to the data being entered by end-user supports validation and modification of this data.  
For additional details refer to the description of the **TListEditingChangeInfo** object.

### Example

```
Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _  
    ByVal objGridCell As TListGridCell, _  
    ByVal objChangeInfo As TListEditingChangeInfo)  
  
    ' Verify that data being entered in column 3 is a valid number  
  
    If GridCell.Col = 3 Then  
        If IsNumeric(objChangeInfo.Value) = False Then  
            Beep  
            ObjChangeInfo.Value = objChangeInfo.OldValue 'restore data  
        End If  
    End If  
End Sub
```

### Remarks

**Note:** Changes, within this event, to the settings of the editing object in use (not changes to the data) will take affect only after the cell's editing is cancelled or completed.

### See Also

**GridCellEditingChange**, **GridCellAfterEditing**, **GridCellEditingKeyDown**, **GridCellEditingKeyUp** and **GridCellEditingKeyPress** events, **EditingMode** and **CellEdit** properties, **TListEditingChangeInfo** object.

---

## GridCellEditingKeyDown, GridCellEditingKeyUp and GridCellEditingKeyPress Events

### Applies to

TListGridCell object

### Description

These events are triggered as a result of a Keyboard actions during editing.

### Syntax

```
Sub TList1_GridCellEditingKeyDown ( [Index As Integer,]  
    ➔ ByVal GridCell As TListGridCell, KeyCode As Integer, Shift As Integer)  
Sub TList1_GridCellEditingKeyUp ( [Index As Integer,]  
    ➔ ByVal GridCell As TListGridCell, KeyCode As Integer, Shift As Integer)  
Sub TList1_GridCellEditingKeyPress ( [Index As Integer,]  
    ➔ ByVal GridCell As TListGridCell, KeyAscii As Integer)
```

### Remarks

KeyCode and Shift parameters are as for the standard KeyUp event. GridCell parameter refers to the cell being edited.

### See Also

GridCellRequestEditing and GridCellAfterEditing events, EditingMode and CellEdit properties

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

---

# GridLinesColor Property

## Applies To

TListGrid object

## Description

Returns or sets a value that determines what color should be used to draw grid lines.

## Syntax

```
TListGridObject.GridLinesColor [= color& ]  
[form.]TList.Grid.GridLinesColor [=color& ]  
[form.]TList.ItemGrid(ItemIndex).GridLinesColor [=color& ]
```

## Remarks

**GridLinesColor** property is used only if **GridLinesStyle** property is set to 1 (Lines). Raised and inset grid lines are always drawn in black and white.

Setting of this property updates control, unless the **Redraw** property is set to False.

## Data Type

Long

---

# GridLinesStyle Property

## Applies To

TListGrid object

## Description

This property returns or sets a value that determines what type of lines should be drawn between cells.

## Syntax

```
TListGridObject.GridLinesStyle [= enum% ]  
[form.]TList.Grid.GridLinesStyle [= enum% ]  
[form.]TList.ItemGrid(ItemIndex).GridLinesStyle [= enum% ]
```

## Settings

The **GridLinesStyle** property setting are:

Setting	Value	Description
TLGRIDLINES_NONE	0	No lines in-between cells.
TLGRIDLINES_HORIZONTAL	1	Horizontal and border lines.
TLGRIDLINES_VERTICAL	2	Vertical and border lines.
TLGRIDLINES_BOTH	3	Vertical, horizontal and border lines. <b>Default.</b>
TLGRIDLINES_INTERNAL_HOR	4	Horizontal lines only.
TLGRIDLINES_INTERNAL_VER	5	Vertical lines only.
TLGRIDLINES_INTERNAL_BOTH	6	Vertical and horizontal lines.

## Remarks

The color of the lines is determined by the **GridLinesColor** property.

---

Note: Three last styles were added to improve grid appearance. Now the combination this property and Grid.BorderStyle one allows the developer to get the full control over the grid appearance.

---

## Data Type



---

## GridLines3DStyle Property

### Applies To

TListGrid object

### Description

This property returns or sets a value that determines what type of 3D lines should be drawn between cells.

### Syntax

```
TListGridObject.GridLines3DStyle [= enum% ]
[form.]TList.Grid.GridLines3DStyle [= enum% ]
[form.]TList.ItemGrid(ItemIndex).GridLines3DStyle [= enum% ]
```

### Settings

The **GridLinesStyle** property setting are:

Setting	Value	Description
TLGRIDLINES3D_NONE	0	Regular 1 pixel lines with no 3D effects.
TLGRIDLINES3D_2COLOR	1	Two-color lines. GridLinesColor and GridLines3DLightColor properties specify the line colors.
TLGRIDLINES3D_3COLOR	2	Three-color lines. GridLinesColor , GridLines3DLightColor and GridLines3DShadowColor properties specify the line colors.

### Remarks

The color of the lines is determined by the **GridLinesColor**, GridLines3DLightColor and GridLines3DShadowColor properties.

### Data Type

Integer

---

## GridLines3DLightColor and GridLines3DShadowColor Properties

### Applies To

TListGrid object

### Description

Returns or sets a value that determines what color should be used to draw grid lines with 3D effect.

### Syntax

```
TListGridObject.GridLines3DLightColor [= color& ]
[form.]TList.Grid.GridLines3DLightColor [=color& ]
[form.]TList.ItemGrid(ItemIndex).GridLines3DLightColor [=color& ]
```

### Remarks

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

These properties would be used only if **GridLines3DStyle** property is set to TLGRIDLINES3D\_2COLOR or TLGRIDLINES3D\_3COLOR values.

### Data Type

Long

---

## GridRowActivate Event / GridRowDeactivate Event

### Applies to

TListGrid object

### Description

The **GridRowActivate** event occurs right after a row becomes active (having focus), but not necessary selected for multi-selection modes) but before any changes are displayed on the screen. The **GridRowDeactivate** event occurs when a row is being deactivated ( losing focus), but before the corresponding changes become visible on the screen.

### Syntax

```
Sub TList_GridRowActivate([Index As Integer], ByVal objGridRow As TListRowDef, ByVal IReserved As Long)
Sub TList_GridRowDeactivate([Index As Integer], ByVal objGridRow As TListRowDef, ByVal IReserved As Long)
```

### Parameters

Parameter	Description
<i>objGridRow</i>	Returns a reference to the row that becomes active or deactivated.
<i>IReserved</i>	Reserved for this version.

### Remarks

The **GridRowActivate** event won't be triggered if the row is not enabled for activating (see **Activatable** property for details) Use the **GridRowClick** events for trapping clicks over disabled rows.

The **ActiveRow**, or **Row** properties are updated after the **GridRowDeactivate** event, but before the **GridRowActivate** event. Thus the previously Active Row can still be referenced via these properties inside the GridRowDeactivate event.

---

**Note:** Upon entry to this event the *objGridRow* parameter will be initially set by TList to *Nothing* if the event was triggered as a result of deleting the corresponding Row object.

---

---

## GridRowTitleClick Event, GridColumnTitleClick Event, GridCornerTitleClick Event


### Applies to

TListGrid object

### Description

These events occurs when the user clicks a cell in a Title Row or Column in any Grid object in TList.

### Syntax

```
Sub TList_GridRowTitleClick([Index As Integer], ByVal ObjTitleCell As TListGridCell,
 ByVal objGridRow As TListRowDef, ByVal Button As Integer)
```

```
Sub TList_GridCornerTitleClick([Index As Integer], ByVal ObjTitleCell As TListGridCell,
    ByVal Button As Integer)
Sub TList_GridColumnTitleClick([Index As Integer], ByVal ObjTitleCell As TListGridCell,
    ➡ ByVal objGridCol As TListColDef, ByVal Button As Integer)
```

## Parameters

Parameter	Description
objTitleCell	Returns a reference to the clicked grid cell object.
objGridRow	Returns a reference to the cell definition object for the row containing the clicked title cell.
objGridCol	Returns a reference to the cell definition object for the column containing the clicked title cell.
<i>Button</i>	Returns an integer identifying which button was pressed. This value is composed of the following bit fields: left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively.

## Remarks

For any of these events, the Row or Column clicked can be identified by the **Row** or **Col** property of the TListGridCell Object passed as a parameter of the event.

Clicking on a Title Cell in a Grid does not generally change the activation state of any Grid Cell.

**Important:** The TList.ActiveGrid, TList.ActiveRow, Grid.Col and Grid.Row properties should not be used within these event – such use may result in unpredicted ( and unsupported) behavior.

## Example

```
' Click on Row Title
Sub TList_GridRowTitleClick( ByVal ObjRowTitleCell As TListGridCell , _
    ByVal objGridRow As TListRowDef, ByVal Button As Integer)
    RowNumber = ObjGridCell.Row
    ItemIndex = objGridCell.Grid.RowToItemIndex(RowNumber)
End Sub

' Click on Column Title
Sub TList_GridColumnTitleClick( ByVal ObjColTitleCell As TListGridCell , _
    ByVal objGridCol As TListColDef, ByVal Button As Integer)
    ColumnNumber = ObjGridCell.Col
    ColumnName = ObjGridCol.ValueName
End Sub

' Click on Corner Title
Sub TList_GridCornerTitleClick( ByVal ObjCornerTitleCell As TListGridCell , _
    ByVal Button As Integer)
    Row = ObjcornerTitleCell.Row
    Col = ObjcornerTitleCell.Col
End Sub
```

# HasCell Property

## Applies To

TListGrid object

## Description

This property determines whether a specified cell exists.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

Not available at design time.

### Syntax

```
Grid.HasCell(Row&, Col&) [=Boolean%]  
TListGridObject.HasCell(Row&, Col&) [=Boolean%]  
[form.]TList.Grid.HasCell(Row&, Col&) [=Boolean%]  
[form.]TList.ItemGrid(ItemIndex).HasCell(Row&, Col&) [=Boolean%]
```

### Remarks

This property is needed because you cannot just compare the return value of TList.**Cell** property to **Null**. If a cell does not exist and the **Cell** property is accessed, the **GridCell** object is created automatically, so **Cell** property always returns a **GridCell** object.

To remove a cell and free up the memory it takes you can set **HasCell** to False. This has the same effect as setting of **Cell** property to **Nothing**.

### Data Type

Boolean

---

## HasGrid Property

### Applies to

TList object

### Description

This property determines whether TList Grid property is set to any TListGrid object.

### Syntax

```
TList1.HasGrid [ = Boolean%]
```

### Remarks

This property is needed because you cannot just compare the return value of TList.Grid property to **Null**. If a grid does not exist and the Grid property is accessed, the Grid object is created automatically, so **Grid** property always returns a Grid object.

To remove a Tree grid you can set **HasGrid** to False. This has the same effect as setting of **Grid** property to **Nothing**.

### Data Type

Object

---

## HasSubItems Property

### Applies to

TList object

### Description

This property indicates whether an item has subordinate items. The **HasSubItems** property is an array whose *index* values correspond to the items in the list.

Not available at design time and Read-Only at run time.

### Syntax

```
[form.]TList.HasSubItems(index&)
```

### Remarks

If an item has subordinate items, the **HasSubItems** property will return True regardless of whether the subordinate items are visible. To determine whether a specific item is visible, use the **IsItemVisible** property.

#### Data Type

Boolean

---

## Height Property

#### Applies to

TList object

#### Description

This property specifies the height of the TList control.

#### Syntax

```
[form.]TList.Height [= numeric_expr]
```

#### Remarks

For more information, see the description of the **Height** property in the *Microsoft Visual Basic On-Line Help*.

#### Data Type

Single

---

## HelpContextID Property

#### Applies to

TList object

#### Description

Determines the context number of the Help topic associated with the TList control. Used to provide context-sensitive Help for your application.

#### Syntax

```
[form.]TList.HelpContextID [= numeric_expr]
```

#### Remarks

For more information, see the description of the **HelpContextID** property in the *Microsoft Visual Basic On-Line Help*.

#### Data Type

Long

---

## HScroll and VScroll Events

#### Applies to

TList object

#### Description

These events are generated whenever the user scrolls TList in a horizontal or vertical direction. They are triggered before scrolling of the control.

#### Syntax

```

Sub TList1_HScroll ( [Index As Integer,] bCancelDefault As Boolean,
    ➡ ByVal ScrollCode As Integer, ByVal Pos As Long)
Sub TList1_VScroll ( [Index As Integer,] bCancelDefault As Boolean,
    ➡ ByVal ScrollCode As Integer, ByVal Pos As Long)

```

## Parameters

bCancelDefault - initialized as False upon entry to the event subroutine, setting the CancelDefault parameter to True before exiting the subroutine will prevent scrolling.

ScrollCode - one of the constants declared in TLIST8.BAS file:

Constant	Value	Description
TL_SB_LINEUP	0	Scroll one line up.
TL_SB_LINEDOWN	1	Scroll one line down.
TL_SB_PAGEUP	2	Scroll one page up.
TL_SB_PAGEDOWN	3	Scroll one page down.
TL_SB_THUMBPOSITION	4	Scroll to absolute position. The current position is specified by the Pos parameter.
TL_SB_THUMBTRACK	5	Drag scroll box (thumb) to specified position. The current position is specified by the Pos parameter.
TL_SB_TOP	6	Scroll to top.
TL_SB_BOTTOM	7	Scroll to bottom.
TL_SB_ENDSCROLL	8	End scroll.

Pos - Specifies the current position of the scroll box if the ScrollCode parameter is TL\_SB\_THUMBPOSITION or TL\_SB\_THUMBTRACK; otherwise, the Pos parameter is not used.

## Example

The following code captures a vertical scroll event and tells TList to ignore scrolling caused by dragging the scrollbar thumbnail until the user releases the scroll thumbnail. This is useful for instance to prevent generation of **ItemQueryData** events during scrolling of a list built with virtual items.

```

Sub TList1_VScroll(bCancelDefault As Boolean, _
    ByVal ScrollCode As Integer, ByVal Pos As Long)

    If ScrollCode = 5 Then CancelDefault = True
End Sub

```

# HWND Property

## Applies to

TList object

## Description

Specifies a window handle of the TList control. Not available at design time; read-only at run time.

## Syntax

```
[form.]TList.hWnd
```

## Remarks

For more information, see the description of the **hWnd** property in the *Microsoft Visual Basic Language Reference*.

### Data Type

Integer

---

## Image Property

### Applies to

TListNode object

### Description

The **Image** (TList object) property is an integer array indexed according to the current setting of the **CurrentIndexMethod** property. Each element of the array corresponds to an item in the TList control and reflects the graphic to be used for an individual non-selected item in the TList control. The **Image** (TListNode Object) property is an ordinal property of corresponding TListNode Object. Note that this property refers to the same picture that you can set via **TList.Image(index)**. Not available at design-time ; read/write at run time.

### Syntax

```
[form.]TList.Image(index&)[= picture]  
TListNode.Image[= picture]
```

### Settings

The **Image** property settings are:

Setting	Description
(none)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. At run time, you can set this property using the <b>LoadPicture</b> function on a bitmap, icon, or metafile.

### Remarks

If the **Image** property is not directly set by code, or is cleared as in setting to an empty picture, {such as by setting to **LoadPicture("")** } , it will be set automatically by the control to an appropriate graphic as defined for the control by the **Picture.... properties** (*PictureClosed*, *PictureRoot*, etc) depending on the state of the item (Closed, Open, Root, etc.) and the setting of the **PictureType** property.

At run time, you can use **Clipboard** methods such as **GetData**, **SetData**, and **GetFormat** with the non-text Clipboard formats CF\_BITMAP, CF\_METAFILE, and CF\_DIB, as defined in **CONSTANT.TXT**, a Visual Basic™ file that specifies system defaults.

Use this property to access the picture of TList items.

Setting of the **Image** property updates the control unless the **Redraw** property is set to False.

Each item having its own individually assigned picture, requires TList to allocate an additional 16 bytes as well as the size of the additional data itself.

At run time, the **Image** property can be set to any other object's **DragIcon**, **Icon**, **Image**, or **Picture** property, or you can assign it the graphic returned by the **LoadPicture** function. The **Image(index)** property can only be assigned directly.

When setting individual images for items in TList, remember that Visual Basic's **LoadPicture** function always returns a reference to a unique image even if loading the same image many times. It is MUCH faster and much more efficient to load the image just once and then reference it repeatedly.

For example:

```
'The following results in 10 distinct images taking up space
For I = 1 to 10
    TList1.Image(I) = LoadPicture (SomePictureFile)
Next
```

The following results in the same list, but only one copy of the image is held in memory:

```
Picture1.Picture = LoadPicture(SomePictureFile)
For I = 1 to 10
    TList1.Image(I) = Picture1.Picture
Next
```

Or TListNode Object can be used instead:

```
Picture1.Picture = LoadPicture(SomePictureFile)
For I = 1 to 10
    TList1.Nodes(I).Image = Picture1.Picture
Next
```

## Data Type

Object

---

# ImageStretch Property

## Applies to

TList object

## Description

This property determines whether the picture in an item will be automatically resized to fit the dimensions specified by the **ItemImageDefHeight** and **ItemImageDefWidth** properties.

## Syntax

```
[form.]TList.ImageStretch [= bool%]
```

## Settings

The **ImageStretch** property settings are:

Setting	Description
True	Picture will be stretched
False	(Default) Picture won't be stretched.

## Remarks

Setting of **ImageStretch** property updates the control unless the **Redraw** property is set to False. Pictures larger than 255 pixels in any dimension will be resized to this maximum limit regardless of the **ImageStretch** setting.

## Data Type

Integer (Boolean)

---

# Indent Property

## Applies to

TListNode object

## Description

TList's **Indent** property is equivalent to MSOutline's **Indent** property. This property allows to change the indent of specified TList item.

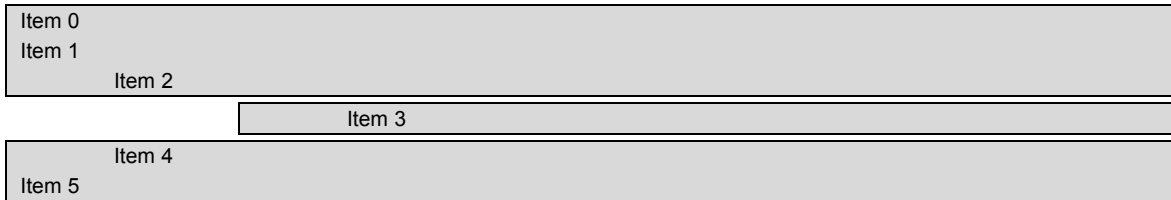


## Syntax

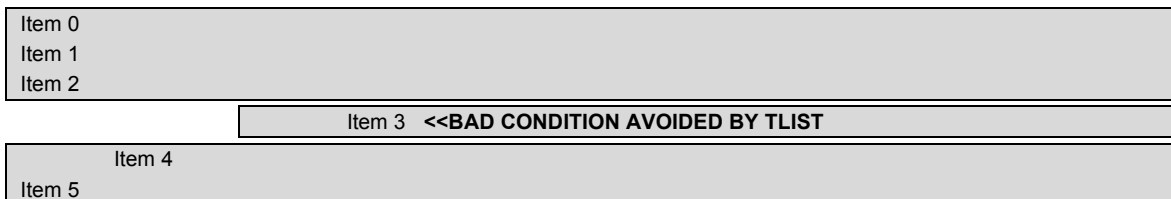
```
[form.]TList.Indent(index&)[ = new_indent%]  
TListNode.Indent[ = new_indent%]
```

## Remarks

TList's **Indent** property is equivalent to MSOutline's **Indent** property with the exception that reducing the indentation of an item with a child will not generate an error as it does with MSOutline.



Now set TList.**Indent**(2) = 0, this would cause an error in MSOutline, TList instead moves the children of Item 2 with Item 2.



The outline after call to TList.**Indent**(2) = 0:



Note the **Indent** property doesn't depend on the **VisualRoot** property settings.

## Data Type

Array(Integer) (TList object)

Integer (TListNode Object)

## See Also

Shift property

---

# Indentation Property

## Applies To

TListLevelDef object

## Description

The **Indentation** property is a long, which reflects an item's hierarchic indentation with which a given TListLevelDef object is linked.

Read-only.

## Syntax

```
[form.]TList.LevelDefs(IndentationLevel&).Indentation
```

## Data Type

Long

---

## Index Property (TListNode Object)

### Applies to

TListNode object

### Description

Returns the index of the node in the parent's sub tree. It is a zero-based index enumerating only direct subordinates of the node's parent (as if CurrentIndexMethod is set to TLSYS\_LEVEL value). Changing this property moves the node to a new position in the parent's tree corresponding to the new index value.

### Syntax

<code>Node.Index[ = Index&amp;] or Index = Node.Index</code>
--

### Data Type

Long

### Example

<p>Starting with</p> <ul style="list-style-type: none"><li>Item 0</li><li>Item 1<ul style="list-style-type: none"><li>Item 1.1</li><li>Item 1.2</li><li>Item 1.3</li><li>Item 1.4</li></ul></li><li>Item 2</li><li>Item 3</li></ul> <p>The following VB code:</p> <pre>Dim CurNode As TListNode Set CurNode = TList1.Node(5) CurNode.Index = 0</pre> <p>Results in:</p> <ul style="list-style-type: none"><li>Item 0</li><li>Item 1<ul style="list-style-type: none"><li>Item 1.4</li><li>Item 1.1</li><li>Item 1.2</li><li>Item 1.3</li></ul></li><li>Item 2</li><li>Item 3</li></ul> <p>The Node "Item 1.4", originally node 5, is moved to the first position (index =0 ) within its parent's list.</p>
--

### Remarks

Specifying an out of range index value will move the node to the end of the sub tree – it will become its parent's last child.

---

## Index Property

### Applies To

TList object

TListColDef object

### Description

**TList** control's **Index** property identifies the control in a control array. Available only if the control is part of a control array; read-only at run time.

**TListColDef** object's **Index** property identifies the object in a **TListColDefs** object collection. Read-only at run time.

### Syntax

```
[form.]TList.Index  
[form.]TList.Grid.ColDefs(ColDefIndex&).Index
```

### Remarks

The index property of a TList control will only have a value if TList is a member of a control array. For more information, see the description of the **Index** property in the *Microsoft Visual Basic On-Line Help*.

The **TListColDef** object's **Index** property valid settings are in the range (0; 2,147,483,647):

### Data Type

Integer

---

## IndexByBM Method

### Applies to

**TList** object

### Description

The **IndexByBM** method returns the index for an item pointed to by its long integer item *bookmark*. The return value is -1 if error occurs.

Not available at design time.

### Declarations

```
[form.]TList.IndexByBM(ByVal hTreeBookmark&) As Long
```

### Example

```
' remove an item that is no longer needed,  
' by reference to its bookmark:  
Ret_Code% = TList1.IndexByBM(hTreeBookmark&)  
TList1.RemoveItem Ret_Code%
```

In the OCX version of TList the VB **IndexByBM** function was replaced with the method with the same name. The VB function can be found in the TLIST8.BAS file.

### See Also

**ItemBM** property and **IsValidBM** method

---

## Insert Property

### Applies to

**TList** object

### Description

This property copies items referenced by a *bookmark* or stored in a *tree buffer*, and adds them as peers immediately before the item whose index is specified.

Not available at design time and write-only at run time.

### Syntax

```
[form.]TList.Insert(index&) = tree_buffer&  
or
```

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

```
[form.]TList.Insert(index&) = Bookmark&
```

### Remarks

Use the **FreeBuffer** method to free memory when you are done with the *tree buffer*.

Setting of the **Insert** property updates the control unless the **Redraw** property is set to False.

When copying from a Bookmark, all of the characteristics of the bookmarked item are retained. Any children of the bookmarked item are also copied and their parent/child relationships maintained.

Note the **Insert** property depends on the **VisualRoot** property settings.

### Data Type

Long

---

## InsertItem Property

### Applies to

TList object

### Description

This property inserts a new item before an item specified by its index. All items that are below this index will be shifted down (their indices will change accordingly). This property is a string array containing index values corresponding to the items in the list.

Not available at design time and write-only at run time.

### Syntax

```
[form.]TList.InsertItem(index&) = string_expr$
```

### Settings

The **InsertItem** property setting is a string of text displayed with the item. See the **List** property to access or change the text after an item has been added.

### Remarks

Setting of **InsertItem** property updates the control unless the **Redraw** property is set to False.

Note the **InsertItem** property depends on the **VisualRoot** property settings.

### Data Type

String

---

## InvBorderStyle Property

### Applies to

TList object

### Description

Specifies whether a cell border must be changed when the cell is selected.

### Syntax

```
[form.]TList.InvBorderStyle[ = enum%]
```

### Remarks

The **InvBorderStyle** property settings are:

Setting	Description
TLINVBORDERSTYLE_NONE	Do not change the border when a cell is selected.
TLINVBORDERSTYLE_CHANGE	(Default) Change the border when a cell is selected. 3D style border is changed to the opposite (inset vs

raised) and 2-D style border is drawn with a color as specified by **SelfForeColor** property.

### Data Type

Integer

---

## InvImage Property or SelectedImage Property

### Applies to

TListNode object

### Description

The **InvImage** or **SelectedImage** (TListNode Object) property defines the image to be displayed for selected items.

Not available at design time; read/write at run time.

### Syntax

```
[form.]TList.InvImage(index&)[ = picture]  
TListNode.SelectedImage[ = picture]
```

### Settings

The **InvImage** or **SelectedImage** property settings are:

Setting	Description
(none)	(Default) No picture
(Bitmap, icon, metafile)	Specifies a graphics. At run time, you can set this property using the <b>LoadPicture</b> function on a bitmap, icon, or metafile.

### Remarks

**InvImage** is an integer array, indexed according to the current setting of the **CurrentIndexMethod** property. Each element of the array corresponds to an item in the TList control and reflects the graphic to be displayed when the specified item in the TList control is selected.

If the **InvImage** property is not directly set by code, or is cleared by setting to an empty picture (such as **LoadPicture("")**), the image displayed will be that defined by the *PictureInverted* property. If *PictureInverted* is also undefined or is cleared, the image will be taken from the **Image** property or the appropriate choice of **PictureRoot**, **PictureLeaf**, or **PictureOpen**.

This property has NO effect unless there is an image also defined for the unselected state by either the **Image**, **PictureRoot**, **PictureLeaf**, or **PictureOpen** properties.

At run time, you can use **Clipboard** methods such as **GetData**, **SetData**, and **GetFormat** with the non-text Clipboard formats CF\_BITMAP, CF\_METAFILE, and CF\_DIB, as defined in *CONSTANT.TXT*, a *Visual Basic* file that specifies system defaults.

Setting of **InvImage** property updates the control unless the **Redraw** property is set to False.

Each item having its own individually assigned picture(s), requires TList to allocate an additional 16 bytes as well as the size of additional data itself.

At run time, the **InvImage** property can be set to any other object's **DragIcon**, **Icon**, **Image**, or **Picture** property, or you can assign it the graphic returned by the **LoadPicture** function. The **InvImage** property can only be assigned directly.

### Data Type

Object

---

## InvStyle Property

### Applies to

TList object

### Description

The **InvStyle** property determines how TList displays selected items.

### Syntax

```
[form.]TList.InvStyle [= enum%]
```

### Settings

The **InvStyle** property settings are:

Setting	Description
0	(Default)Text and Picture will be inverted when item is selected.
1	Picture will be inverted
2	Text will be inverted
3	The entire line is inverted.
4	The text portion of the selected item is inverted, and the inverted background rectangle color is extended to include the background of the picture.
5	The text portion of the selected item is inverted and the inverted background rectangle color is extended to the right most extent of the line.

### Remarks

The view of the selected item is affected by the setting of **ViewStyle** property.

Setting of **InvStyle** property updates the control unless the **Redraw** property is set to False.

### Data Type

Integer

---

## IsClipboardAvailable Property

### Applies to

TList object

### Description

The IsClipBoardAvailable property returns a true/false value indicating whether there is TList data in the clipboard. Read only at run-time, not available at design time.

### Syntax

```
[form.]TList.IsClipboardAvailable
```

### Example

If compatible data is available, then paste it before the selected item

```
If TList_Dest.IsClipboardAvailable then  
    TList_Dest.Clipboard(TList.ListIndex) = 4  
End If
```

### Data Type

Boolean

---

## IsItemVisible Property

### Applies to

TList object

### Description

This property returns whether an item is currently visible, in other words, whether an item is either currently displayed within the control window or can be viewed by scrolling. The **IsItemVisible** property is an array whose *index* values correspond to the items in the list.  
Not available at design time and read-only at run time.

### Syntax

```
[form.]TList.IsItemVisible(index&)
```

### Data Type

Boolean

---

## IsPrinting Method

### Applies To

TListReport Object

### Description

The **IsPrinting** method returns True if TList is currently printing.

### Syntax

```
[form.]TList.Report.IsPrinting() As Boolean
```

---

## IsValidBM Method

### Applies to

TList object

### Description

The **IsValidBM** method returns a value (True or False) indicating whether the specified item *bookmark* is valid.  
Not available at design time.

### Syntax

```
[form.]TList.IsValidBM(ByVal hTreeBookmark&) As Integer
```

### Example

```
Ret_code = TList1.IsValidBM (hTreeBookmark&)
```

In the OCX version of TList this function was replaced with Visual Basic function with the same name. This function can be found in the TLIST8.BAS file.

### See Also

**ItemBM** property and **IndexByBM** method

---

## IsValidBuffer Method

### Applies to

TList object

### Description

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

This method determines whether the passed *tree buffer* value is valid. Returns True or False.  
Not available at design time.

### Syntax

```
[form.]TList1.IsValidBuffer(ByVal hTreeBuffer As Long) As Integer
```

### Remarks

The parameter *hTreeBuffer* is a variable of a *Long* type.

---

You can use any TList control to check any *tree buffer*.

---

## Item Method (TListSelectedGridCells, TListSelectedGridColumn and TListSelectedGridRows Objects)

### Applies to

TListSelectedGridRows object

TListSelectedGridColumn object

TListSelectedGridCells object

### Description

This method returns a reference to the object from the a collection that holds a series of TListGridCell, TListColDef and TListRowDef objects representing all selected cells/columns or rows.

### Syntax

```
[TListGridCell] = [TListGrid].SelectedCells.Item(ByVal IIndex As Long)  
[TListColDef] = [TListGrid].SelectedColumns.Item(ByVal IIndex As Long)  
[TListRowDef] = [TListGrid].SelectedRows.Item(ByVal IIndex As Long)
```

### Values

The method's parameters are:

Parameter	Description
<b>IIndex</b>	An index of the object in the collection.

### Details

Attempting to access an object using incorrect index will trigger an error.

### Example

a) Display all selected cells

```
Dim objSelectedCells as TListSelectedGridCells  
Dim objCell as TListGridCell, ICnt as Long  
Set objSelectedCells = TList1.Grid.SelectedCells  
Debug.Print "Selected Cells"  
For ICnt = 0 To objSelectedCells.Count-1  
    Set objCell = objSelectedCells.Item(ICnt)  
    Debug.Print "objCell =(" & objCell.Row & "," & objCell.Col & ")"  
Next
```

b) Display all selected rows

```
Dim objSelectedRows as TListSelectedGridRows  
Dim objRowDef as TListRowDef, ICnt as Long  
Set objSelectedRows = TList1.Grid.SelectedRows
```



```

Debug.Print "Selected Rows"
For ICnt = 0 To objSelectedRows.Count-1
    Set objRowDef = objSelectedRows.Item(ICnt)
    Debug.Print "Row Index =" & objRowDef.Index
Next

```

### c) Display all selected columns

```

Dim objSelectedColumns as TListSelectedGridColumns
Dim objColDef as TListColDef, ICnt as Long
Set objSelectedColumns = TList1.Grid.SelectedColumns
Debug.Print "Selected Columns"
For ICnt = 0 To objSelectedColumns.Count-1
    Set objColDef = objSelectedColumns.Item(ICnt)
    Debug.Print "ColDef Index =" & objColDef.Index & objColDef.ValueName
Next

```

## Item Property (TVWSelectedNodes Object)

### Description

This property returns an Node object specified by a given index.

Note: the valid index is a long number in range between 0 and Count-1.

Read-only.

### Syntax

```
Set objNode = [form.]TListTreeView.SelectedNodes.Item(Index)
```

### Data Type

Node object

## ItemActivate Event / ItemDeactivate Event

### Applies to

TList object

### Description

The **ItemActivate** event occurs after an item becomes active ( gets the focus)

Syntax

```

Sub TList_ItemActivate([Index As Integer], ByVal ItemIndex As Integer, ByVal IReserved As Long)
Sub TList_ItemDeactivate([Index As Integer], ByVal ItemIndex As Integer, ByVal IReserved As Long)

```

### Parameters

Parameter	Description
<i>ItemIndex</i>	Returns an index of the item that becomes activate.
<i>IReserved</i>	Reserved for this version.

### Remarks

An activated item is not necessarily selected.

The ItemActivate event won't be generated if the item is not enabled for activating (see **Activatable** property for details). The **ItemClick** or **Click** events may be used to trap clicks over disabled items.

The **ListIndex** property is updated after the **ItemDeactivate** event, but before the **ItemActivate** event. Thus the previously Active Item can still be referenced using this property inside the ItemDeactivate event.

---

*Note:* Upon entry to this event *ItemIndex* parameter will be initially set by TList to *Nothing* if the event was triggered as a result of deleting the corresponding item.

---

## ItemAlwaysHidden Property

### Applies to

TList object

### Description

This property Specifies specifies whether an item is displayed if all its parent items are expanded and the **ShowHiddenItems** property is False.

See Setting below for more detailed description.

Not available at design time.

### Syntax

```
[form.]TList.ItemAlwaysHidden (index&)[=Integerbool%]
```

### Settings

The **ItemAlwaysHidden** property settings are:

Setting	Description
False	(Default) Item is visible if all parent items of this item are expanded. This item subordinates are visible if the item is expanded.
True	Item is invisible regardless this item parents expanding state. This item subordinates are never visible too.

### Remarks

If a ParentItem is always hidden, its children can never be seen regardless of its expand state or their **ItemAlwaysHidden** state.

### Remarks

Boolean

---

## ItemBackColor and ItemForeColor Properties

### Applies to

TList object

### Description

The **ItemBackColor** property determines the background color of an item.

The **ItemForeColor** property determines the foreground color used to display text in an item.

These properties are long arrays whose *index* values correspond to the items in the list. Not available at design time; read/write at run time.

### Syntax

```
[form.]TList.ItemBackColor(index&)[ = color&]  
[form.]TList.ItemForeColor(index&)[ = color&]
```

### Remarks

The default settings are determined by **BackColor** and **ForeColor** properties.

Setting of these properties update the control unless the **Redraw** property is set to False.

An additional 16 bytes of memory are required for each item having its own assigned Fore or **BackColor**.

### Data Type

Long(Array)

### See Also

**BackColor** and **ForeColor** properties

---

## ItemBM Property

### Applies to

**TList** object

### Description

The **ItemBM** array contains LongInteger pointers to an item specified by its index. The bookmark associated with an item will not change even if the items index changes as a result of adding items earlier in the list or changing the **CurrentIndexMethod**. See also **IndexByBM** and **IsValidBM** methods. Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.ItemBM(Index&)
```

### Data Type

Long(Array)

---

## ItemCell Property

### Applies to

**TList** object

### Description

This property returns a TListCellDef object, which provides control over item cell display. Read-only, not available at design time.

### Syntax

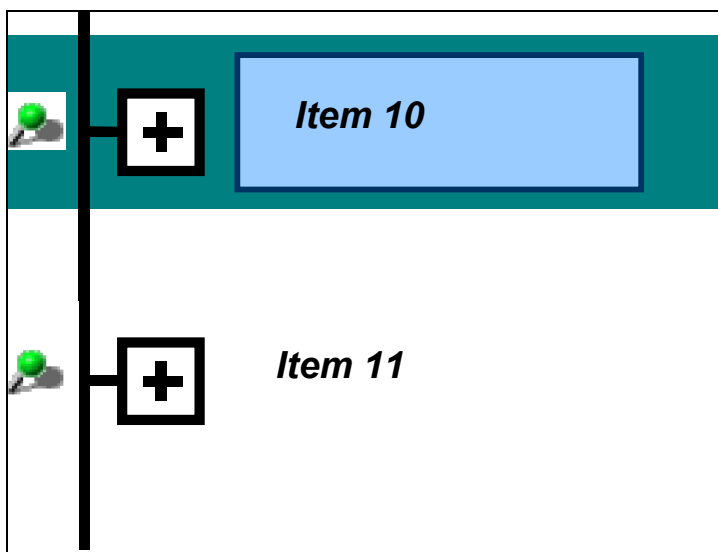
```
[form.]TList.ItemCell (index&)
```

### Example

The code below shows how to set a background color for an item cell (shown in the figure below as the area surrounding the text "item 10") different from the item's overall background color and default color (shown as white).

```
TList1.BackColor = RGB(255,255,255) ' White  
TList1.ItemBackColor(10) = RGB(0,127,0) ' Green  
TList1.ItemCell(10).BackColor = RGB(0,0,127) ' Blue
```

The result is:



### Data Type

Object

## ItemCheckboxValue Property

### Applies to

TListCheckBox object

### Description

Returns or sets a value that determines the state of a checkbox if an item is checked (has a checkmark next to it) for a List / Tree item or first column of a grid.

### Syntax

```
[form.]TList.ItemValues(ByVal ItemIndex&) [= Variant]
```

### Data Type

Variant

### Remarks

**ItemCheckboxValue** determines the item checkbox state according to **CheckedValue/UncheckedValue/GrayedValue** properties. The **ItemCheckboxValue** is automatically updated in response to end-user actions (mouse-click / or doubleclick) on the item if **Editable** property is switched on.

The checkbox state for grid cells is equivalently read or set using the **CheckBoxValue** property.

### Example:

```
TList1.AddItem "Item1"
TList1.ItemCell(0).EditInfo.Style = TLEDITINFO_CHECKBOX           'Show checkbox
TList1.ItemCell(0).EditInfo.EditInfo = TLEDITABLE_ALWAYS_ONCLICK 'Edit on click
TList1.ItemCell(0).EditInfo.CheckBox.CheckedValue = "Yes"        "'Yes" string as checked value
TList1.ItemCell(0).EditInfo.CheckBox.UncheckedValue = "No"       "'No" string as unchecked value
TList1.ItemCheckboxValue(0) = "Yes"                               'Now the item is checked
```

### See Also:

**CheckboxValue**, **EditInfo**, **Editable** properties and **TListCheckBox** object

---

## ItemClick Event

### Applies to

TList object

### Description

This event occurs when the user selects an item in TList control by clicking the mouse button.

### Syntax

```
Sub TList_ItemClick([Index As Integer], ByVal ItemIndex As Long,  
    ➡ ByVal Button As Integer)
```

### Remarks

This event passes *ItemIndex*, the *index* of the item in the list that was clicked.

The *Button parameter* returns an integer identifying which button was pressed. This value is composed of the following bit fields: left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. **Note:** neither ItemClick nor ItemDbClick events are fired for items belonging to a Grid except for the Tree column in a TreeGrid. For other cells use the GridCellClick, GridCellDbClick, GridColumnTitleClick, GridRowTitleClick or GridCornerTitleClick events.

---

---

## ItemDbClick Event

### Applies to

TList object

### Description

This event occurs when the user double clicks an item in TList control by clicking the mouse button.

### Syntax

```
Sub TList_ItemDbClick([Index As Integer], ByVal ItemIndex As Long,  
    ➡ ByVal Button As Integer)
```

### Remarks

This event passes *ItemIndex*, the *index* of the item in the list that was double clicked.

The *Button parameter* returns an integer identifying which button was pressed. This value is composed of the following bit fields: left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively.

---

**Note:** neither ItemClick nor ItemDbClick events are fired for items belonging to a Grid except for the Tree column in a TreeGrid. For other cells use the GridCellClick, GridCellDbClick, GridColumnTitleClick, GridRowTitleClick or GridCornerTitleClick events.

---

---

## ItemEditText Property

### Applies to

TList object

### Description

Setting the **ItemEditText** property enables, concludes or cancels an in-place editing operation.

Upon setting this property to TLEDITMODE\_BEGIN, the **RequestEditing** event will be triggered.

Upon setting this property to TLEDITMODE\_END or TLEDITMODE\_CANCEL, the **AfterEditing** event will be triggered.

Not available at design-time, write only at run-time.

## Syntax

```
[form.]TList.ItemEditText(index&) = %enum
```

## Settings

The **ItemEditText** property settings are:

Name	Setting	Description
TLEDITMODE_CANCEL	0	Cancel editing of the item text.
TLEDITMODE_END	1	Conclude editing of the item text.
TLEDITMODE_BEGIN	2	Start editing of the item text.
TLEDITMODE_CONTINUE	3	Continue editing This setting may be used within the AfterEditing event.

## Data Type

Integer

## Note:

The index parameter of the ItemEditText property is used only to initiate editing (when setting the property to 2 ). Otherwise (when canceling, completing, or resuming editing) , the index parameter is ignored.

If it is necessary to continue editing (for example, after a wrong value is entered by an end user), the setting **ItemEditText** (0) = 3 can be added within the **AfterEditing** for any item that is being edited event.

## See Also

**RequestEditing**, **AfterEditing**, **EditingKeyDown**, **EditingKeyUp**, and **EditingKeyPress** events and description of In-place Editing in the *Using TList* chapter

---

# ItemEditingChange Event

## Applies to

**TListCheckBox** object

**TListComboBox** object


**TListTextBox** object

## Description

This event is triggered in response to any end-user action (keyboard or mouse) that occurs while the user is editing Tree or List items using a built-in editor (**TListTextBox**, **TListComboBox**, **TListCheckBox**, etc) takes place.

The event may be used to validate the data being entered by end-user and correct the data if necessary during editing.

## Syntax

```
Sub TList_ItemEditingChange (ByVal ItemIndex As Long,  
                              ByVal objChangeInfo As TListEditingChangeInfo)
```

## Parameters

Parameter	Description
ItemIndex	Index of the item being edited.
ObjChangeInfo	<b>A TListEditingChangeInfo</b> object providing access to the

	data being edited, and supporting validation and modification of this data. For additional details refer to the description of the <b>TListEditingChangeInfo</b> object.
--	---

### Example

```
' Type Ahead Assistance
Private Sub TList1_ItemEditingChange(ByVal ItemIndex As Long, _
    ByVal objChangeInfo As TListEditingChangeInfo)

' For editing with Combobox object,
' help the end-user by comparing what he types with combobox list items
' and completing the typing for him ( with the untyped portion selected).

' Check type of cell, only act if typing with combobox object
If objChangeInfo.ValueType = TLED_CHANGE_COMBOBOX_EDITAREA
    ' compare text user has typed with items from the list,
    ' substitute it with a item text from the list if there is a match
    Dim strUserEnteredData As String
    StrUserEnteredData = objChangeInfo.Value ' string typed by user
    Dim iDataLen as Long
    iDataLen = Len( StrUserEnteredData ) ' length of string typed by user
    Dim objItem as TListComboltem
    For Each objItem In objChangeInfo.EditInfoObject.ComboBox.Items
        If Left$(objItem.CellValue, iDataLen) = StrUserEnteredData _
            Then ' match found – set full text of list item in text area
                objChangeInfo.Value = objItem.CellValue
                ' select the portion of the text we have just added
                objChangeInfo.SelStart = iDataLen
                objChangeInfo.SelLength = Len(objChangeInfo.Value)-iDataLen
            Exit For
        End If
    Next
End If
End Sub
```

### Remarks

Note: Changes, within this event, to the settings of the editing object in use, (not changes to the data) will take affect only after the cell's editing is cancelled or completed.

### See Also

**GridCellEditingChange**, **GridCellAfterEditing**, **GridCellEditingKeyDown**, **GridCellEditingKeyUp** and **GridCellEditingKeyPress** events, **EditingMode** and **CellEdit** properties, **TListEditingChangeInfo** object.

---

## ItemFont... Properties

### Applies to

**TList** object

### Description

When set, these properties take precedence over the standard font characteristic properties (**FontBold**, **FontItalic**, etc) in determining the font style for an item. Setting the properties to True

or False sets font characteristic accordingly. **ItemFontName** may be set as a string to any installed font. **ItemFontSize** may be set to an integer value.

These properties are integer arrays whose indexed values correspond to the items in the list.

Not available at design time; read/write at run time.

### Syntax

```
[form.]TList.ItemFontBold(index&)[ = boolean%]  
[form.]TList.ItemFontItalic(index&)[ = boolean%]  
[form.]TList.ItemFontStrike(index&)[ = boolean%]  
[form.]TList.ItemFontUnder(index&)[ = boolean%]  
[form.]TList.ItemFontName(index&)[ = string_expression$]  
[form.]TList.ItemFontSize(index&)[ = numeric_expression$]  
- each sets or returns the font appropriately.
```

### Settings

The **ItemFontBold**, **ItemFontItalic**, **ItemFontStrike**, **ItemFontUnder** properties' settings are:

Setting	Description
True	Turns on formatting in that style.
False	Turns off formatting in that style.

### Remarks

Setting of these properties update the control, unless the **Redraw** property is set to False.

An additional 16 bytes of memory are required for each item having its own assigned font.

The **ItemFontSize** property is measured in points (1/72 inches), with a maximum value of 2048 points.

### Data Type

Boolean (Array) - for Bold, Italic, Strike, and Under

String (Array) - for Name

Integer (Array) - for Size

### See Also

**FontName**, **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties

---

## ItemGrid Property

### Applies to

TListGrid object

### Description

Returns a TListGrid object, which provides an information about a Grid which the specified item owns.

Read-only, not available at design time.

### Syntax

```
[form.]TList.ItemGrid (index&)
```

### Example

The code below assigns an item grid to a Grid object

```
Dim Grid As TListGrid  
Set Grid = TList1.ItemGrid(nParentIndex)
```

The code below creates an item grid with 3 columns and 2 rows:

```
TList1.ItemGrid(10).Cols = 3
```



```
TList1.ItemGrid(10).Rows = 2
```

### Data Type

TListGrid Object

---

## ItemIndex Property

### Applies To

TListValue Object

### Description

Returns an index of the tree item to which this **TListValue** object is linked. Read-only

### Syntax

```
[form.]TList1.ItemValues(ItemIndex&, ValueName&).ItemIndex
```

### Data Type

Long

---

## ItemIndexToRow Method

### Applies To

TListGrid Object

### Description

Converts the item index of an item into a corresponding Grid row index. If the item index doesn't match any row of the grid, -1 is returned.

### Syntax

```
TListGridObject.ItemIndexToRow(ByVal ItemIndex As Long) As Long
```

---

## ItemHasGrid Property

### Applies to

TListGrid object

### Description

Determines whether an item has a grid.  
Not available at design time.

### Syntax

```
[form.]TList. ItemHasGrid (index&) [=Bool!%]
```

### Remarks

This property is needed because you cannot just compare the return value of TList.**ItemGrid**(Index&) property to **Null**. If a grid doesn't exist and *Grid* property is accessed the *Grid* object is created automatically, so **ItemGrid** property always returns a *Grid* object. To remove an item grid you can set **ItemHasGrid** to False. This has the same effect as setting of the **ItemGrid** property to **Nothing**.

### Data Type

Boolean

---

## ItemHeight Property

### Applies to

TList object

### Description

Returns or sets the height of a given item. Not available at design time.

### Syntax

```
[form.]TList.ItemHeight(ItemIndex&) [= number&]
```

### Settings

The **ItemHeight** property settings are:

Setting	Description
-1	(Default) The item height will be calculated.
> -1	Specifies the height of an item

### Remarks

Using following statement `[form.]TList.ItemHeight(-1) = Value&` sets specified height value for all tree items.

### See Also

**RowHeight** property

---

## ItemImageDefWidth and ItemImageDefHeight Property

### Applies to

TList object

### Description

These properties determine the width and height of item's picture when the **ImageStretch** property is set to True. Otherwise setting of these properties has no visual effect but the new values will be used next time the **ImageStretch** property is set to True.

### Syntax

```
[form.]TList.ItemImageDefWidth [= setting&]  
[form.]TList.ItemImageDefHeight [= setting&]
```

### Remarks

These properties settings are:

Setting	Description
0	(Default) Picture in an item is displayed in its actual size.
> 0	The size of an item's picture is expressed in terms of the container's scale. By default measured in twips

Setting of these properties update the control unless the **Redraw** property is set to False.

### Data Type

Long

---

# ItemMark Property

## Applies to

TList object

## Description

TList supports the assigning of a MARK to elements in the outline. There is an array of marks which can hold up to 255 members. Each Mark has an associated Tag and Picture. Setting the **ItemMark** property specifies the **MarkTag** and **MarkPicture** to be linked to that item. Not available at design time.

## Syntax

```
[form.]TList.ItemMark(ItemIndex&) [= MarkIndex&]
```

## Example

```
Sub TList_Click()  
    'Get some text from Mark Array based on which item is clicked  
    Item_Clicked& = TList.ListIndex  
    TagNumber& = TList1.ItemMark(Item_Clicked&)  
    GetText$ = TList1.MarkTag(TagNumber&)  
End Sub
```

## Data Type

Long

---

# ItemMultiLine Property

## Applies to

TList object

## Description

This property specifies whether an item can accept and display multiple lines of text. This property is an array in which each item refers to the subordinate item. See Settings below for more detailed description.

Not available at design time.

## Syntax

```
[form.]TList.ItemMultiLine(index&)[=Integer%]
```

## Settings

The **ItemMultiLine** property settings are:

Name	Value	Description
TL_MULTILINE_DEFAULT	0	(Default) The value of the <b>DefMultiLine</b> property determines whether to wordwrap the specified item.
TL_MULTILINE_OLD_TRUE	-1	TList allows multiple lines of text, and word wraps long lines of text according to the <b>WidthOfText</b> property setting. Old version's constant.
TL_MULTILINE_TRUE	1	TList allows multiple lines of text, and word wraps long lines of text according to the <b>WidthOfText</b> property setting.
TL_MULTILINE_FALSE	2	TList ignores carriage returns and restricts data to a single line.

## Remarks

Setting of **ItemMultiLine** property updates the control unless the **Redraw** property is set to False.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

The TList control wraps text to a maximum line width specified by the **WidthOfText** property. If the **WidthOfText** property set to zero, the item will be word wrapped to fit within the client area of TList.

---

## ItemParent Property

### Applies to

TList object

### Description

This property returns an index of the parent of an item specified by its index.  
Read-only, not available at design-time.

### Syntax

```
[form.]TList.ItemParent (item&)
```

### Example

```
' Get the index of the parent of currently selected item.  
ParentIndex& = TList1.ItemParent(TList.ListIndex)
```

### Data Type

Long

### See Also

ItemPrevSibling, ItemNextSibling properties

---

## ItemParentBM Property

### Applies to

TList object

### Description

This property returns a Bookmark for the parent of an item specified by its index.  
Read-only, not available at design-time.

### Syntax

```
[form.]TList.ItemParentBM(item&)
```

### Example

```
' Get the index of the parent of currently selected item.  
ParentIndex& = TList1.IndexByBM(TList.ItemParentBM(TList.ListIndex))
```

### Data Type

Long

---

## ItemPMPicType Property

### Applies to

TList object

### Description

This property determines whether TList forces the display of a plus or minus image next to an item. This property can be used to override the default TList behavior and, for example, show a plus or minus picture next to an item which doesn't have any children.  
Run-time only.

## Syntax

```
[form.]TList.ItemPMPicType(item&) [= enum%]
```

## Settings

The **ItemPMPicType** property settings are:

Setting	Description
0	(Default) TList will show the appropriate picture depending on the state of an item.
1	Always show plus picture regardless the number of children of an item.
2	Always show minus picture regardless the number of children of an item.
3	Do not show any picture.

## Data Type

Integer

---

# ItemPrevSibling, ItemNextSibling, and ItemLastSubItemIndex Properties

## Applies to

TList object

## Description

The **ItemNextSibling** property always returns the next sibling of an item, i.e., the next child of the same parent.

Likewise the **ItemPrevSibling** property will return the previous sibling of an item. A value of -4096 indicates an item has no previous or next sibling.

The **ItemLastSubItemIndex** property returns the index of the last subordinate item for a specified item.

Read-only, not available at design-time.

## Syntax

```
[form.]TList.ItemPrevSibling (item&)  
[form.]TList.ItemNextSibling (item&)  
[form.]TList.ItemLastSubItemIndex (item&)
```

## Data Type

Long

## See Also

**ItemParent** property

---

# Item Property

## Applies To

TListValues Object

TListLevelDefs Object

TListColDefs Object

TListPages Object

## TListNodes Object

### Description

This property returns an object specified by a given index. For example, for the **TListColDefs** object collection, this property returns a reference to a **TListColDef** object stored in this collection. Read-only.

### Syntax

```
[form.]TList1.LevelDefs.Item(LevelDefItemIndex&)  
[form.]TList1.Grid.ColDefs.Item(ColDefItemIndex&)  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs.Item(ColDefItemIndex&)  
[form.]TList1.ItemValues(ItemIndex&).Item(ValueNameIndex)  
[form.]TList1.Report.Pages.Item(Page)  
[form.]TListNodes.Item(Page)  
This property is default for all these objects so we can rewrite these as follows:  
[form.]TList1.LevelDefs (LevelDefItemIndex&)  
[form.]TList1.Grid.ColDefs (ColDefItemIndex&)  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs (ColDefItemIndex&)  
[form.]TList1.ItemValues(ItemIndex&, ValueNameIndex)  
[form.]TList1.Report.Pages(Page)
```

### Data Type

Long

---

## ItemQueryData Event

### Applies to

TList object

### Description

This event is triggered whenever TList needs to display or retrieve data from a virtual item. (for example when a parent having virtual children is expanded). It is also triggered when the **RefreshItems** method is called.

### Syntax

```
Sub TList_ ItemQueryData([Index As Integer],  
    ➡ ByVal ItemIndex As Long,  
    ➡ ByVal SiblingIndex As Long)
```

### Remarks

This event passes *ItemIndex*, the *index* of the item whose data is required.

*SiblingIndex* is the index of this item within parent's children list (only direct children are enumerated).

*ItemIndex* is the actual Index of the item within the Tree.

For example, if item13 has 6 virtual children, this event will be called 6 times when item 13 is expanded, the first time *ItemIndex* will be 14 and *Siblingindex* will be 0. The next time *ItemIndex* will be 15 (depending on whether item 14 itself has children) and *SiblingIndex* will be 1. Finally the event will be called with *ItemIndex* 19 and *SiblingIndex* 6.

### Example

```
Sub TList1_ItemQueryData(ByVal ItemIndex As Long, _  
    ByVal SiblingIndex As Long)  
    TList1.List(ItemIndex) = "Virtual Kid " & SiblingIndex  
    TList1.ItemForeColor(ItemIndex) = RGB(127, 0, 0)  
End Sub
```

---

## Items Property (TListComboBox Object)

### Applies to

TListComboBox object

### Description

References the collection of combobox list items - **TlistComboItem** objects

The Items collection has methods for managing the list of combobox items: adding, removing, etc.

Each **TlistComboItem** object represents a combobox list item.

Each

### Syntax

```
[TListCellDef].EditInfo.ComboBox. Items
```

### Data Type

TListComboItems object

### Example

```
' Reference items belonging to combobox set up for all grid cells
With TList1.Grid.GridCellDef.EditInfo.ComboBox
    .Items.Add 10, Picture1.Picture, "Ten"
    .Items.Add 20, Picture2.Picture, "Twenty"
    ...
EndWith
' Create TListComboItem belonging to first column of a grid
With TList1.Grid.Coldefs(1).CellDef.editinfo.ComboBox
    Dim tlici as TlistComboItem
    Set tlici = .Items.Add 20, Picture2.Picture, "Twenty"
    tlici.BackColor = RGB (100, 200, 0)
    ...
EndWith
```

### Remarks

See TListComboItems Object

---

## ItemSorted, ItemSortingMethod, ItemSortingStyle, and ItemSortingKey Properties

### Applies to

TList object

### Description

These properties specify how TList sorts root items or a specified branch of the tree.

- The **ItemSorted** property initiates, halts or resets the sorting.
- The **ItemSortingMethod** property specifies ascending or descending sort order.
- The **ItemSortingStyle** property provides additional control such as Numeric or Case Sensitive Sorting.
- The **ItemSortingKey** property specifies what data (visible text or hidden ItemValues) should be used as the key for the sorting.

### Syntax

```
TList.ItemSorted(ByVal ParentIndex as Long)[ = enum%]  
TList.ItemSortingMethod(ByVal ParentIndex as Long, ByVal SortPriority as Variant)[ = enum%]
```

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

```
TList.ItemSortingStyle(ByVal ParentIndex as Long, ByVal SortPriority as Variant) [= settings%]  
TList.ItemSortingKey(ByVal ParentIndex as Long, ByVal SortPriority as Variant) [= ValueName$]
```

### Syntax Change:

- **ItemSorted** constants have been changed.
- An empty or unassigned **ItemSortingKey** value previously indicated sorting by the old (obsolete) **ItemXXXValue** properties. In TList 8, an empty or unassigned **ItemSortingKey** indicates sorting by the visible text. To sort by an **ItemXXXValue** property set the **ItemSortingKey** to "\_\_ObsoleteValue".
- **ItemSortingMethod** introduced to specify Ascending or Descending sorting order.
- Data may be sorted by a prioritized list of Multiple Columns or ValueNames, each with it's own sorting style specified by **ItemSortingStyle** property.

### Parameters

Name	Description
ParentIndex	Index value pointing to parent of the items to be sorted. To sort roots, set ParentIndex to -1.
SortPriority	(default 0) – identifies sort priority for the specified ValueName Multiple sorting styles, sorting keys and sorting methods may be specified. TList will sort data first according to specification of ItemSortingMethod, ItemSortingKey, and ItemSortingStyle having parameter SortPriority = 0, followed the sort criteria data having SortPriority 1, 2, 3, ...

### ItemSorted property settings:

Setting	Constant	Description
0	TLSORTSTATE_OFF	(Default) Sort OFF – Sorting is off. Additions or modifications to data in TList will not affect the order of the list, nor will changes to the sort properties.
1	TLSORTSTATE_ON	Sort On – Sorting is On. When first set TList performs an immediate sorting using current sort settings. Any changes to data or to sorting properties will immediately be reflected in a new sorting order.
2	TLSORTSTATE_RESET	Reset – Reset the sorting state – all sorting settings for a specific branch will be removed, and <b>ItemSorted</b> will be automatically set back to 0

### ItemSortingMethod property settings:

Setting	Constant	Description
0	TLSORT_ASCENDING	(Default) Sort items in the ascending order.
1	TLSORT_DESCENDING	Sort items in the descending order.

### ItemSortingKey property settings:

The **ItemSortingKey** property is a Variant which should be either Empty (""), or a any ValueName for which ItemValues have been assigned to TList.



If **ItemSortingKey** is Empty(""), sorting will be based on the Visible Text held within TList (or for grid objects if there are multiple columns then on the default column of TList - the column holding the main Tree / List data ).

If **ItemSortingKey** contains a valid ItemValue name or numeric column index for grid objects, sorting will be based on the ItemValues referenced by the ValueName or held in the Grid Column (Grid.Coldef object) named by that ValueName.

If **ItemSortingKey** contains the special value "\_\_ObsoleteValue", TList will sort by the data held in the old ItemXXXvalue properties (ItemStrValue, ItemIntValue...).

---

Sorting may be performed on multiple columns in prioritized fashion by specifying multiple ItemSortingKeys, each with a SortPriority parameter.

---

### ItemSortingStyle property settings

The **ItemSortingStyle** property should be set as the sum of the desired Bit Flag values from the following table:

Constant	Value	Description
TL_SORT_IGNORE_CASE	&H1	If set, ignore case while sorting.
TL_SORT_NUMBERS_SORT	&H2	If set, treat string representation of numbers as numbers. "10" comes after "2"
TL_SORT_STRING_SORT	&H4	If set, treat punctuation the same as symbols
TL_SORT_GROUP_ATTOP	&H8	If set, TList groups all items which have children at the top of the sorted sub-tree. <b>Note:</b> This Flag is processed by TList only for settings of ItemSortingStyle with 2nd parameter (SortPriority ) = 0
TL_SORT_GROUP_ATBOTTOM	&H10	If set, TList groups all items which have children at the bottom of the sorted branch. <b>Note:</b> This Flag is processed by TList only for settings of ItemSortingStyle with 2nd parameter (SortPriority ) = 0
TL_SORT_IGNORE_KANATYPE	&H20	Do not differentiate between Hiragana and Katakana characters. Corresponding Hiragana and Katakana characters compare as equal.
TL_SORT_IGNORE_NONSPACE	&H40	Ignore non-spacing characters
TL_SORT_IGNORE_SYMBOLS	&H80	Ignore symbols.
TL_SORT_IGNORE_WIDTH	&H100	Do not differentiate between a single-byte character and the same character as a double-byte character.
TL_SORT_FLOAT_NUMS_SORT	&H200	Apply for sorting of Floating point numbers. Takes fractional part of number into account when sorting.

---

**Note:** Bit Flag values, TL\_SORT\_GROUP\_ATTOP and TL\_SORT\_GROUP\_ATBOTTOM may not be applied together.

---

### Tips:

- To sort all items in a flat List or Grid, or to sort the root items of a Tree, specify a parent index of **-1** for the **ItemSorted** property, or set the **.Sorted** property of **Leveldefs(0)** object.
- To fully sort the entire data set, set the **ItemSorted** property for each item which has children, or set the **.Sorted** property of each **LevelDef** object.
- When working with a Grid, sorting may be controlled using the **SortingKey**, **SortingMethod**, **SortingStyle**, and **.Sorted** properties.

### Data Type

**ItemSorted** – Enum Integer

**ItemSortingKey** - String

**ItemSortingMethod** – Enum Integer

**ItemSortingStyle** –Integer

### Example

#### 1. Sort root items by visible text

```
' – Ignore Case, keep items with children at the top

' Identify branch to sort
ParentIndex = -1 ' special index of -1 refers to parent of full data set

' Reset Sort Criteria and Turn off Sorting until new Criteria Set
TList1.ItemSorted(ParentIndex) = TLSORTSTATE_RESET

' Specify Sort Criteria
TList1.ItemSortingKey (ParentIndex) = "" ' not required, it is default
TList1.ItemSortingMethod (ParentIndex) = TLSORT_ASCENDING
TList1.ItemSortingStyle (ParentIndex)= TL_SORT_IGNORE_CASE _
+ TL_SORT_GROUP_ATTOP

' Sort Now
TList1.ItemSorted(ParentIndex) = TLSORTSTATE_ON
```

#### 2. Specify sorting of branch – by visible text and then by hidden numeric data

```
TList1.ParentIndex = 12 ' – will sort subitems in branch of 12th root
TList1.ItemSorted (ParentIndex) = TLSORTSTATE_RESET

' Specify First Sort Priority
TList1.ItemSortingKey (ParentIndex,0) = "" ' not required, it is default
TList1.ItemSortingMethod (ParentIndex,0) = TLSORT_ASCENDING
TList1.ItemSortingStyle (ParentIndex,0) = TL_SORT_IGNORE_CASE

' Specify Second Sort Priority
TList1.SortingKey (ParentIndex,1) = "SSNumber" ' item value points at num data
TList1.SortingMethod (ParentIndex,1) = TLSORT_ASCENDING
TList1.SortingStyle (ParentIndex,1) = TL_SORT_NUMBERS_SORT

TList1.ItemSorted(ParentIndex) = TLSORTSTATE_ON
```

#### 3. Sort children of 1<sup>st</sup> root of tree by the data assigned to valuenam "ss number"

```
TList1.ItemSorted(0) = TLSORTSTATE_RESET
TList1.ItemSortingMethod (0,0) = TLSORT_ASCENDING
TList1.ItemSortingKey(0, 0) = "SS Number"
TList1.ItemSorted( 0 ) = TLSORTSTATE_ON
```

- 
4. **Sort root items by visible data first, then by a value name "first name" and finally by a value name "last name"**

```
TList.ItemSorted(-1) = TLSORTSTATE_RESET  
TList.ItemSortingMethod (1,0) = TLSORT_ASCENDING  
TList.ItemSortingKey(-1, 0) = "__Tree"  
TList.ItemSortingKey(-1, 1) = "First Names"  
TList.ItemSortingKey(-1, 2) = "Last Names"  
TList.ItemSorted(-1) = TLSORTSTATE_ON
```

### See Also

**Sorted** property, **SortingStyle** property, **SortingKey** property, **SortingMethod** property for **TListGrid** object.

---

## ItemTag Property

### Applies to

**TList** object

### Description

The **ItemTag** property accesses an associated hidden data element for each TList item.  
Not available at design time.

### Syntax

```
[form.]TList.ItemTag(index&)[=Variant]
```

### Example

The sample below demonstrates how to set values for items:

```
For I = 1 to 10 ' Create 10 items  
    TList1.AddItem "Item" & I  
    ' -3 can be used instead of TList1.NewIndex  
    TList1.ItemTag(-3) = Rnd(I)  
Next I
```

---

Note that there is only one tag value per item but it can be of any type.

---

The code below sets and gets values of various types for 4 items:

```
TList1.ItemTag(0) = I% ' Set Integer value  
TList1.ItemTag(1) = L& ' Set Long value  
TList1.ItemTag(2) = S! ' Set Single value  
TList1.ItemTag(3) = L$ ' Set String value  
Set TList1.ItemTag(4) = PictureBox.Picture ' Set Picture value  
I% = TList1.ItemTag(0) ' Retrieve Integer value  
L& = TList1.ItemTag(1) ' Retrieve Long value  
S! = TList1.ItemTag(2) ' Retrieve Single value  
L$ = TList1.ItemTag(3) ' Retrieve String value  
Set PictureBox.Picture = TList1.ItemTag(4) ' Retrieve Picture value
```

### Remarks

Using TList with the **ItemTag** property is an easy way to create an array of images, which can then be easily saved to a file.

If you need more than one piece of hidden data with an item use the **ItemValues** property, but if you only need a single additional value associated with each item, the **ItemTag** property is the simplest solution.

---

Note that there was a bug in Visual Basic 5.0; it is not possible to assign an object to an array property.

---

The following code will trigger an error in Visual Basic 5.0, but will execute properly in Visual Basic 4.0:

```
Dim rsTableContents As RecordSet
...
Set TList1.ItemTag(l) = rsTableContents
```

Below is a work-around of this bug:

```
Dim rsTableContents As RecordSet
...
Dim VarTmp
Set VarTmp = rsTableContents
TList1.ItemTag(l) = VarTmp
```

### Data Type

Variant

---

## ItemToolTip and ToolTip Properties

### Description

These properties reference a TListToolTip object, the properties of which (**.Text**, **.Picture**, **.Style**) may be read or set, and specify the content and presentation of the tooltip window to be displayed when the **.ToolTipsMode** property is set to TLTOOLTIPSMODE\_ENABLE and the mouse is left over the text for some time (as specified by ToolTipsDelay property).

### Applies to

**TList** object  
**TListGrid** object  
**TListGridCells** objects

### Syntax

```
TList.ItemToolTip(index)  
TList.ToolTip  
TList.Grid.ToolTip  
TList.Grid.Cells(row, col).ToolTip
```

### Example

The sample below demonstrates how to set values for items:

```
Dim tToolTip as TListToolTip  
Set tToolTip as TList1.ToolTip  
' OR  
' Set tToolTip as TList1.ItemToolTip(ItemIndex).....  
' Set tToolTip as TList1.Grid.ToolTip.....  
' Set tToolTip as TList1.Grid.Cells(RowIndex, CellIndex).ToolTip.....  
tToolTip.Text = "This is a tooltip"  
tToolTip.Picture = Picture1.Picture  
tToolTip.Style.BackColor = RGB ( 100,100,200)
```

### See Also

TListToolTip object

---

## Item...Value Properties

[Applies to](#)

[TList object](#)

[Description](#)

The **Item...Value** and **ItemType** used in prior editions of TList still exist in TList, and point to another distinct hidden data element. But these properties are now considered to be obsolete and should not be used.

---

## ItemValues and ItemHasValue Properties (Values property of TListNode Object)

[Applies to](#)

[TList object](#)

[TListNode object](#)

[Description](#)

The **ItemValues** and **Values** properties holds named data associated with a TList item.

The **ItemHasValue** property determines whether data with a given name exists. It also can be used to remove data specified by a name data.

Not available at design time.

[Syntax](#)

```
[form.]TList.ItemValues(ByVal Itemindex&)  
[form.]TList.ItemValues(ByVal Itemindex&, ByVal ValueName As Variant)  
TListNode.Values(ByVal ValueName As Variant)  
[form.]TList.ItemHasValue(ByVal Itemindex&, ByVal ValueName As Variant) [= bool%]
```

[Remarks](#)

TList supports the storage of associated data with each element in the list. By default this associated data is not displayed, but it may be displayed in columns by setting of the **ValueName** property of the **ColDef** object. The associated data may be any Visual Basic data types. If more than one piece of data is associated with an item, use the **ItemValues** property, but if you have to keep only a single additional value, then use of the **ItemTag** property is the simplest solution.

When used with a single parameter, **ItemValues** property returns a TListValues object which is a collection of TListValue objects.

When used with 2 parameters, **ItemValues** property returns a TListValue object, which is a container for one piece of additional data.

When displaying Grids within TList, the values in the grid cells may also be referred to by the ItemValues array. Default names assigned to grid cell columns are: "\_\_RowTitle", "\_\_Tree", "T2", "T3", "T4", etc. Setting the ColDef.ValueName property to the name assigned to an ItemValue will cause TList to display the appropriate ItemValues within that column of a grid.

[Example](#)

---

**Note** You may read from this property without explicately referencing the Value property:

```
TList1.ItemValues(100, "LastName")  
'Is equivalent to  
TList1.ItemValues(100, "LastName").Value
```

---

But, you have to use the full form to assign a value:

```
TList1.ItemValues(100, "LastName").Value = "Fred"
```

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

To assign some associated data with item 100:

```
TList1.ItemValues(100, "LastName").Value = "Fred"  
TList1.ItemValues(100, "FirstName").Value = "Henry"  
TList1.ItemValues(100, "Age").Value = 59
```

Rewritten with using TListNode Objects:

(note: it is faster method since there is no necessity looking for the item for every assignment)

```
Dim TLNode As TListNode  
Set TLNode = TList1.Node(100)  
TLNode.Values("LastName").Value = "Fred"  
TLNode.Values("FirstName").Value = "Henry"  
TLNode.Values("Age").Value = 59
```

To assign the value 666 to all associated data for item 100:

```
Dim TListValues1 As TListValues  
'TList1.ItemValues(100) returns a TListValues object  
TListValues1 = TList1.ItemValues(100)  
Dim X As TListValue  
For Each X In TListValues1  
    X = 666 ' Here you can use short form not X.Value.  
Next
```

Rewritten in shorter form:

```
Dim X As TListValue  
For Each X In TList1.ItemValues(100)  
    X = 666 ' Here you can use short form not X.Value.  
Next
```

This code removes all associated data:

```
Dim X As TListValue  
For Each X In TList1.ItemValues(100)  
    TList1.ItemHasValue(100, X.ValueName) = False  
Next
```

## Data Type

Object

---

# ItemVirtualParent, ItemVirtualCount, VirtualParent and VirtualCount Properties

## Applies to

TList object

TListNode object

## Description

The **ItemVirtualParent** and **VirtualParent** property returns a True/False value indicating whether an item's children are Virtual.

The **ItemVirtualCount** and **VirtualCount** property determines how many virtual children a given item has.

Not available at design time.

## Syntax

```
[form.]TList.ItemVirtualParent(ByVal Itemindex&) [= bool%]  
[form.]TList.ItemVirtualCount(ByVal Itemindex&) [= VirtualItemCount%]  
or  
TListNode.VirtualParent [= bool%]  
TListNode.VirtualCount [= VirtualItemCount%]
```

## Example

Below is a sample, which adds a thousand virtual subordinates for an item:

```
TList1.ItemVirtualCount(2)= 1000  
Or  
TList1.Node(2).VirtualCount = 1000
```

## Remarks

TList supports virtual items and nested virtual items. Virtual items are items which are not always kept in memory. The necessary data is requested by TList as needed using the **ItemQueryData** event, and is removed from memory when no longer needed (for instance when the virtual item's parent item is collapsed).

The **ItemQueryData** event is generated each time TList needs data to display a virtual item (as when a virtual item's parent is expanded).

If **ItemVirtualCount** property is set for an item which has **ItemVirtualParent** property set to False, the **ItemVirtualParent** property is automatically reset to True.

If **ItemVirtualCount** is set to a value less than the number of virtual children already in memory, excess children are removed from the list.

If the number of children for a given parent is not known at the time the parent is added to the list, one approach may be to initialize the ItemVirtualCount for that parent to 1, and then to reset this value within the Expand event of TList.

\* \* Note that the ItemVirtualCount property can not be reliably changed inside the ItemQueryData event.

## See Also

The **ItemQueryData** event

---

# ItemUrl Property

## Applies to

[TList object](#)

## Description

TList's Web Auto Navigation feature uses relative URLs stored in **ItemUrl** property to navigate to a web site when a user double-clicks on an item. See **WebAutoNavigate** property for details.

Not available at design time.

## Syntax

```
[form.]TList.ItemUrl(ByVal Itemindex&) [= String$]
```

## Example

Below is a sample which points to a very interesting web page:

```
TList1.ItemUrl(2)= "http://www.bennet-tec.com/common/whoarewe.htm"
```

## Data Type

String

---

# HitTest Method

## Applies to

[TList object](#)

## Description

This method determines what graphical TList object is under the mouse cursor.

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

## Syntax

```
[form.]TList.HitTest(  
    ➡ ByVal X As Single, ByVal Y As Single,  
    ➡ TargetObject As Variant) As Long
```

## Settings

The **HitTest** method return value is one of the following:

Setting	Value	Description
TLHITTEST_ERROR	0	<b>HitTest</b> failed or hits empty space on the control. TargetObject is not set.
TLHITTEST_ITEM	1	<b>HitTest</b> hits an item. TargetObject gets long index of the item.
TLHITTEST_ITEMTEXT	2	<b>HitTest</b> hits item text. TargetObject gets long index of the item.
TLHITTEST_ITEMCELLPICTURE	3	<b>HitTest</b> hits an item cell picture. TargetObject gets long index of the item.
TLHITTEST_ITEMPICTURE	4	<b>HitTest</b> hits a picture. TargetObject gets long index of the item.
TLHITTEST_ITEMPMPICTURE	5	<b>HitTest</b> hits the plus minus picture. TargetObject gets long index of the item.
TLHITTEST_ITEMMARK	6	<b>HitTest</b> hits the mark picture. TargetObject gets long index of the item.
TLHITTEST_GRIDLINES	7	<b>HitTest</b> hits a grid line. TargetObject gets a reference to TListGridCell which owns the line.
TLHITTEST_GRIDCELL	8	<b>HitTest</b> hits a cell. TargetObject gets a reference to TListGridCell.
TLHITTEST_GRIDCELLPICTURE	9	<b>HitTest</b> hits a cell picture. TargetObject gets a reference to TListGridCell.
TLHITTEST_GRIDCELLTEXT	10	<b>HitTest</b> hits a cell text. TargetObject gets a reference to TListGridCell.

## Example

```
Sub TList1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
    Dim TargetObject  
    WhatsHit = TList1.HitTest(X, Y, TargetObject )  
    Select Case WhatsHit  
        Case 1, 2  
            Label1.Caption = "Item " & TList1.List( TargetObject )  
        Case 7, 8, 9, 10  
            Label1.Caption = "Cell " & TargetObject.CellDef.Text  
    End Select  
End Sub
```

---

# KeyboardActivation Property (TList object)

## Applies to

TList object

## Description



This property controls how the user navigates (moves the focus specifying the Active Cell) through the Tree Grid structure using the keyboard.

## Syntax

[TList].KeyboardActivation = [long]
-------------------------------------

## Settings

The **KeyboardActivation** property is a bit-field valued property set by OR'ing the desired combination of flag values:

Constant	Setting	Description
TL_KBDACTIV_NONE	0	(Default setting) The Up and Down arrows keys move the focus through the whole TREE / GRID structure. In this mode Right and Left arrow keys open/close a Tree branch if necessary (dependant on <b>ExplorerCompatible</b> property) and then function as Down and Up arrow keys. So while pressing Up and Down keys user visits all sub-branches and parent items if they are accessible (opened, etc). The tab key may be used to switch focus out of TList to another control on the form except as modified by other flag settings.
TL_KBDACTIV_TAB	&H1	The TAB key is enabled to navigate among Grid Cells and items in a Tree. The focus is moved to the next active item, cell, or row if the user presses the TAB key, and the previous item if the user presses SHIFT+TAB combination. Note: the TAB key will work together (not substitute) with the Right, Left, Up, and Down arrow keys.
TL_KBDACTIV_LEAVEONTAB	&H2	The focus leaves the TList control when the user presses TAB while the focus is on the last cell in the grid or the last open item in the tree, or when the user presses SHIFT+TAB while the first cell/item is active.
TL_KBDACTIV_LEVEL	&H4	Navigation by Tree Level. . The focus is restricted to items/ grid cells at the same hierarchic level as the current active cell or item when navigating with the keyboard. The Up and Down arrow keys will move the focus only inside the current hierarchy level of items / grids. TAB keystrokes move the focus from the last cell in an ItemGrid to the next item or first cell in the next TreeGrid if one exists at the same hierarchy in the tree structure. SHIFT+Tab will move backward in the same manner Note: Using LEFT / RIGHT arrow keys are similarly restricted by level.
TL_KBDACTIV_ACTIVEGRIDONLY	&H8	Applied only for movement within a grids. Moves the focus (active cell/row) only within the

		active grid object; if there are any subordinate grids or items the focus will not pass through them.
--	--	---

### Example

```
TList.KeyboardActivation = TL_KBDACTIVE_TB Or TL_KBDACTIV_LEAVEONTAB
```

### See Also

**ActivationMode** property, **Activable** property

---

## KeyDown and KeyUp Events

### Applies to

**TList object**

### Description

These events occur when the user presses (**KeyDown**) or releases (**KeyUp**) a key while a TList control has the focus. (To interpret ANSI characters, use the **KeyPress** event.)

### Syntax

```
Sub TList_KeyDown([Index As Integer], KeyCode As Integer,  
    ➡ Shift As Integer)  
Sub TList_KeyUp([Index As Integer], KeyCode As Integer,  
    ➡ Shift As Integer)
```

### Remarks

Setting the KeyCode = 0 during the **KeyDown** event will prevent end user navigation through the tree using the cursor keys.

For more information, see the description of the **KeyDown** and **KeyUp** events in the *Microsoft Visual Basic Language Reference*.

---

## KeyPress Event

### Applies to

**TList object**

### Description

This event occurs when the user presses and releases an ANSI key.

### Syntax

```
Sub TList_KeyPress([Index As Integer], KeyAscii As Integer)
```

### Remarks

For more information, see the description of the **KeyPress** event in the *Microsoft Visual Basic Language Reference*.

---

## KeyboardNavigateWhileEditing Property(TListEditInfo)

### Applies to

TListEditInfo object

### Description

This new property controls how TList reacts to navigation keys while the user is editing.

## Syntax

```
.EditInfo.KeyboardNavigateWhileEditing = enum%
```

## Settings

The **KeyboardNavigateWhileEditing** property may be set to one or composed from a logical (**OR**) combination of the following bit flags:

Setting	Value	Description
TL_EDITNAVIGATION_OFF	0	The navigation is OFF.
TL_EDITNAVIGATION_DEFAULT	-1	This indicates that the behavior should be inherited from higher level CellDef. For instance, when specified for a grid cell, the behavior may be inherited from column, row, or overall grid settings.
TL_EDITNAVIGATION_ON_ARROW_KEYS	1	TList stops editing current cell and moves to, and activates, next cell (up/down/left/right one) when user presses one of the arrow keys. At end or beginning of a grid row TList moves to next/previous row.
TL_EDITNAVIGATION_ON_TAB_KEY	2	TList stops editing current cell and moves to, and activates, next cell (left/right) when the user presses either the TAB key or the SHIFT+TAB combination. At end or beginning of a grid row TList moves to next/previous row.

## Example

```
With TList1.Grid
  .Cols = 4
  .Rows = 5
  .ActivationMode = TL_ACTIVMODE_CELL
  .SelectionMode = TL_SELMODE_SINGLE
  With .GridCellDef
    .Selectable = TL_SEL_DISABLED
    With .EditInfo
      .Editable = TLEDITABLE_ALWAYS_ONACTIVATE
      .KeyboardNavigateWhileEditing = _
        TL_EDITNAVIGATION_ON_ARROW_KEYS _
        OR TL_EDITNAVIGATION_ON_TAB_KEY
    End With
  End With
End With
```

## See Also

Editable property

---

# Left Property

## Applies to

## TList object

## Description

Determines the distance between TList and the left edge of its container.

## Syntax

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

```
[form.]TList.Left [ = numeric_expr ]
```

### Remarks

For more information, see the description of the **Left** property in the *Microsoft Visual Basic Language Reference*.

### Data Type

Single

---

## LeftMargin, TopMargin, RightMargin, and BottomMargin Properties

### Applies To

TListReport object

### Description

These properties specify the region on a page where TList will print data. These properties are measured in twips.

Not available at design-time.

### Syntax

```
TListReportObject.LeftMargin [ = twips&]  
TListReportObject.TopMargin [ = twips&]  
TListReportObject.RightMargin [ = twips&]  
TListReportObject.BottomMargin [ = twips&]  
[form.]TList.Report.LeftMargin [ = twips&]  
[form.]TList.Report.TopMargin [ = twips&]  
[form.]TList.Report.RightMargin [ = twips&]  
[form.]TList.Report.BottomMargin [ = twips&]
```

### Data Type

Long

---

## LevelDefs Property

### Applies to

TList object

### Description

**LevelDefs** is an array property of TListLevelDef objects used to specify default settings for all items of a given hierarchic indentation level (0 to 255). Most font and color attribute properties may be applied to the LevelDefObject.

### Syntax

```
[form.]TList.LevelDefs(Index As Short) [ = TListLevelDef Object ]
```

### Settings

Returns an object of TListLevelDef type. The example below sets background color for all items with indentation 4 to green color:

```
TList1.LevelDefs(4).BackColor = QBColor(2)
```

---

## List Property

### Applies to

324 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX

## TList object

### Description

The **List** property is a string array. Each element of the array contains the text associated with and displayed for corresponding items in the TList control.

---

Note that this same text is also used in determining the **FullPath** property and in setting or reading the **CurrentParent** property.

---

Not available at design-time; read/write at run time.

### Syntax

```
[form.]TList.List(index&)[ = string_expression$]
```

### Remarks

Setting the **List** property visibly updates the control unless the **Redraw** property is set to False. Elements of the **List** are indexed as specified by the current setting of the **CurrentIndexMethod** property.

**TList.Text** is equivalent to **TList.List(TList.ListIndex)**.

### Data Type

String (Array)

---

## ListCount Property

### Applies to

TList object

### Description

Depending on the method of indexing currently in use (see the **CurrentIndexMethod** property), the **ListCount** property specifies:

1. total number of items in a list, for first method of indexing.
2. total number of visible items in a list for second method of indexing. (Visible items are items whose parents are expanded and whose **ItemAlwaysHidden** property is set to False.
3. total number of items in the *current parent* for third method of indexing.

Not available at design time and read-only at run time.

### Syntax

```
[form.]TList.ListCount
```

### Data Type

Long

---

## ListCountEx Property

### Applies to

TList object

### Description

**ListCountEx** returns the number subordinates of a given item.

Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.ListCountEx(index&)
```

### Remarks

Note **ListCountEx** calculates **all** subordinates not just direct children.

If Index is set to -1, the **ListCountEx** property returns the total number of items in the list (as defined by the **CurrentIndexMethod** property). In this case the result is the same as reading the **ListCount** property.

### Data Type

Long

---

## ListIndex Property

### Applies to

TList object

### Description

The property determines the *index* of the currently selected item in the control.

With the **MultiSelect** property set to True, **ListIndex** sets or returns the index of the item having the focus rectangle.

Not available at design time, read-write at run-time.

### Syntax

```
[form.]TList.ListIndex[ = index&]
```

### Settings

The **ListIndex** property settings are:

Setting	Description
-1	(Default) Indicates that no item is currently selected. Note - It is NOT possible to set <b>ListIndex</b> to this value, a value of -1 may only be read prior to any programmatic or end-user selection.
≥ 0	A number indicating the index of the currently selected item.

### Remarks

You can use this property to set the focus rectangle to the item at the specified index in a multiple-selection mode. The item may or may not be selected in this case.

Setting the **ListIndex** property resets the **CurrentParent** to point to the parent of the **ListIndex** item.

Setting of the **ListIndex** property updates the control, unless the **Redraw** property is set to False.

### Data Type

Long

---

## LoadAndAdd Property

### Applies to

TList object

### Description

This property loads Tree data from a file, adding it as a subordinate to the item pointed to by Index. Write only at run-time, not available at design time.

### Syntax

```
[form.]TList.LoadAndAdd(index&) = FileHandle%
```

### Remarks

If you specify index of -1 the Tree data are added to the end of the list (as defined by the **CurrentIndexMethod** property).

---

Note

- Tree data from several TLists or *tree buffers* may be stored in a single file. The data is stored sequentially
  - The **LoadAndAdd** property depends on the **VisualRoot** property settings.
- 

### Example

```
Dim FreeHandle%
FreeHandle% = FreeFile
' Open file for input.
Open "e:\MyTListFile.tlt" For Binary Access Read As FreeHandle%
TList1.LoadAndAdd(-1) = FreeHandle%
Close FreeHandle% ' Close file.
```

---

## LoadAndInsert Property

### Applies to

### TList object

### Description

The property loads Tree data from a file, adding it as a peer immediately before the item pointed to by the specified Index.

Write only at run-time, not available at design time.

### Syntax

```
[form.]TList.LoadAndInsert(index&) = FileHandle%
```

### Remarks

If you specify index of -1 the Tree data are added to the end of the list.

---

Note

- Tree data from several TLists or *tree buffers* may be stored in a given file. The data is stored sequentially
  - The **LoadAndInsert** property depends on the **VisualRoot** property settings
- 

### Example

```
Dim FreeHandle%
FreeHandle% = FreeFile
' Open file for input.
Open "e:\MyTListFile.tlt" For Binary Access Read As FreeHandle%
TList1.LoadAndInsert(-1) = FreeHandle%
Close FreeHandle% ' Close file.
```

---

## LoadBuffer Method

### Applies to

### TList object

### Description

The method loads tree data from a file to a *tree buffer*. Several *tree buffers* may be stored in the same file. The data is stored sequentially so to get to the third buffer, load and discard the first two. The return value is zero if the function is successful. Otherwise, the return value is an error code. Not available at design time.

### Syntax

```
[form.]TList1.LoadBuffer(hTreeBuffer&, ByVal nFile As Integer) As Integer
```

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Example

```
Dim Dummy%, hTreeBuffer%, FreeHandle%

FreeHandle% = FreeFile

' Open file for input.
Open "c:\MyTListFile.tlt" For Binary Access Read As FreeHandle%
...
Dummy% = TList1.LoadBuffer(hTreeBuffer, FreeHandle%)

if Dummy% <> 0 then ..... ' error
...
Close FreeHandle% ' Close file
```

---

## LoadData Method

### Applies to

### TList object

### Description

This property loads TList properties and data from a file compatible with the **SaveData** method. **LoadData** method loads not only TList's tree structure (items), but TList's properties settings like **FontItalic**, **ViewStyle** etc.

Write only at run-time, not available at design time.

### Syntax

```
[form.]TList.LoadData(FileName As String) As Integer
```

### Remarks

The **LoadData** method returns one of the following values:

Setting	Description
0	Succeeded.
1	Read File Error.
2	File Not Found.
4	Bad format of file.

---

Files loaded by **LoadData** can be created either by **SaveData** method or other TList's **SaveXXX** properties. But files created with **SaveData** can be loaded back ONLY by **LoadData** method.

---

Files created by TDesigner application can be loaded with **LoadData** method.

All TList properties will be changed after using the LoadData method to load from a file, to the property settings to those specified in TList when the SaveData method was used.

---

## LoadData Method (TListTreeView)

### Description

This property loads TListTreeView properties and data from a file compatible with the TListTreeView **SaveData** method. **LoadData** method loads not only TListTreeView property settings but also the whole tree structure.

Not available at design time.

### Syntax

```
[form.]TListTreeView.LoadData(FileName As String) As Integer
```



## Remarks

The **LoadData** method returns one of the following values:

Value	Description
0	Succeeded.
1	Read File Error.
2	File Not Found.
4	Bad format of file.
5	Can't connect references to images with corresponding physical images from within specified ImageList control. Note: the full tree structure will be loaded even if some image references can't be connected with corresponding image.

All TListView properties (except ImageList ) will be changed after using the LoadData method to load from a file, to the property settings to those specified in TListView when the SaveData method was used.

---

**Note:** Since ImageList is not saved in TLV file it is the user's responsibility to provide correct instance of ImageList control for loading tree structure.

User can specify ImageList control either before or right after calling LoadData method.

If an ImageList is not available or doesn't contain the corresponding image when the TLV file is loaded, an item won't display the image but the image reference will be preserved anyway.

If ImageList exists but contains different images for a given ImageList index then that different image will be shown.

**Warning:** TListView LoadData method can only load files created by the TListView SaveData method, not by the SaveData method of the TList object or any save properties of the TList object.

---

## LoadPicture Method

### Applies to

TList object

### Description

This methods loads picture from any external file (note: only BMP, ICO, WMF, EMF formats are supported) and returns reference on it. This reference can be used for initializing any TList picture property. **Note:** caller of this method is responsible for releasing this reference.

Read only at run-time, not available at design time.

### Syntax

<code>[form.]TList.LoadPicture(FileName As String) As Picture</code>
--

## LostFocus Event

### Applies to

TList object

### Description

This event occurs when the TList control loses focus.

### Syntax

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

```
Sub TList_LostFocus([Index As Integer])
```

### Remarks

For more information, see the description of the **LostFocus** event in the *Microsoft Visual Basic On-Line Help*.

---

## MarginLeft, MarginTop, MarginRight, MarginBottom Properties

### Applies to

TListCellDef object

### Description

These properties specify an offset between the cell boundaries and any text or graphics contained within an cell. Measurement is in units of TList's container's Scale Mode.

### Example

```
TListGrid.Cells( row, col).CellDef.MarginLeft = offset  
TListGrid.Cols( column).CellDef.MarginLeft = offset
```

---

## MarkClick and MarkDbClick Events

### Applies to

TList object

### Description

These events are generated when a mark picture associated with an item is clicked or double clicked.

### Syntax

```
Sub TList1_MarkClick ( [Index As Integer,] ByVal I As Long)  
Sub TList1_MarkDbClick ( [Index As Integer,] ByVal I As Long)
```

### Remarks

This event passes *I*, the *index* of the item in the list whose mark picture was clicked or double-clicked.

---

## MarkedItemsAlwaysHidden Property

### Applies to

TList object

### Description

The **MarkedItemsAlwaysHidden** property is an array of 256 elements (0:255) containing boolean values.

The **MarkedItemsAlwaysHidden** property specifies whether items whose **ItemMark** property is equal to a given Mark are always hidden when the **ShowHiddenItems** property is False (regardless of the expand state of the tree).

Not available at design time.

### Syntax

```
[form.]TList. MarkedItemsAlwaysHidden(MarkIndex&)[= long%]
```

### Remarks

If a parent item is AlwaysHidden, its children can never be seen regardless of its expand state or their **ItemAlwaysHidden** state.

---

**Note:** This property should be set to either 0 ( False) or -1 ( True)

---

### Data Type

Long

---

## MarkPicture Property

### Applies to

### TList object

### Description

The **MarkPicture** property is an array containing up to 256 distinct pictures.

The **MarkPicture** property specifies the picture associated with a given Mark. A Mark can be associated with a given TList item using the **ItemMark** property.

MarkPictures are displayed along the far left of the control lined up horizontally with the list elements they are associated with. Typically, MarkPictures are used to visually categorize items. Not available at design time.

### Syntax

```
[form.]TList.MarkPicture(MarkIndex&)[= picture%]
```

### Remarks

The **MarkPicture** is only visible when the **ViewStyleEx** property is set to 2 or 3. Under these conditions, and when **Redraw** is set to True, setting the **MarkPicture** property will update the display of each item which associated with the mark to show the new image.

A default **MarkPicture** may be assigned using the **PictureMark** property.

---

Note The **PictureMark** property and **MarkPicture**(0) refer to the same picture. If you change **PictureMark**, it means that you also change the first item of **MarkPicture** array.

---

### Example

The following example shows how to set the mark picture of TList items:

```
TList1.MarkPicture(MarkIndex&) = LoadPicture("PIC.BMP")
TList1.ItemMark(5) = MarkIndex&
TList1.ItemMark(505) = MarkIndex&
TList1.ItemMark(1005) = MarkIndex&
```

---

## MarkTag Property

### Applies to

### TList object

### Description

The **MarkTag** property is an array property of 256 elements (0:256) containing string values.

The **MarkTag** property specifies the text associated with a given Mark. A Mark can be associated with a given TList item using the **ItemMark** property.

Not available at design time.

### Syntax

```
[form.]TList.MarkTag(MarkIndex&)[=Text$]
```

## Example

The following example shows how to get the tag of the mark associated with an item:

```
MarkIndex& = TList1.ItemMark(1005)
MarkTag$ = TList1.MarkTag(MarkIndex&)
```

---

# MarkWidth and MarkHeight Properties

## Applies to

TList object

## Description

These properties specify the screen display size of mark pictures. If **MarkWidth** property is set to 0, then the width of the default mark is used. If **MarkHeight** property is set to 0, then the height of the default mark is used. The defaults are based on the dimensions of the picture held in the **PictureMark** property. These properties are measured in terms of the units of TList's container. Not available at design time.

## Syntax

```
[form.]TList.MarkWidth[= Width&]
[form.]TList.MarkHeight[= Height&]
```

## Default Value:

15 pixels

## Remarks

Keep in mind that these properties determine the size of all mark pictures. If you have **MarkHeight** and **MarkWidth** properties set to zero, and the **PictureMark** property doesn't have a picture, then the height and width of all mark pictures will be 0 and no mark pictures will be displayed.

---

# MaxLength Property (TListTextBox Object)

## Applies to

TListTextBox objects

## Description

This property sets the maximum number of characters supported for data entry (typing) within a data cell.

## Notes

If **MaxLength** = 0 the Edit window is created in Read-Only mode.

If the text held by the data cell already exceeds the maximum length when editing is initiated, the end-user may delete characters from the cell but may not add additional characters until the string length is less than **MaxLength**.

When text is pasted into a cell, TList will check the length of the text in the clipboard and will truncate the inserted text to limit the length of data in the cell as specified by the **MaxLength** property.

## Syntax

```
[TListCellDef].EditInfo.TextBox.MaxLength [= maximum&]
```

## Data Type

Long

---

## Min and Max Properties (TListDateTime Object)

### Applies to

TListCellDef **object**

### Description

These properties set the earliest and latest values for the DateTime edit object.

### Syntax

```
[TListCellDef].EditInfo.DateTime.Max [= variant]  
[TListCellDef].EditInfo.DateTime.Min [= variant]
```

### Data Type

Variant

### Example:

```
TList.Grid.cells(3,1).celldef.EditInfo.datetime.Min = cDate("1/1/00")  
TList.Grid.cells(3,1).celldef.EditInfo.datetime.Max = cDate("12/31/00")
```

### Remarks

If an end-user tries to select a date beyond min or max settings, the min or max value will be displayed in the date/time editor.

---

## Min and Max Properties (TListSpin Object)

### Applies to

TListCellDef **object**

### Description

These properties set the maximum and minimum values for the **TListSpin** edit object.

### Syntax

```
[TListCellDef].EditInfo.Spin.Max [= maximum&]  
[TListCellDef].EditInfo.Spin.Min [= minimum&]
```

### Data Type

Variant

### Remarks

The behavior of the spin control is reversed if the Spin **Min** property exceeds **Max**.

---

## MinHeight, MaxHeight and HeightScale Properties (TListTextBox Object)

### Applies to

TListTextBox **object**

### Description

The **MinHeight** and **MaxHeight** properties set the minimum and maximum vertical size of the Edit window in pixels. As TList supports scrolling within the edit window these properties do not limit the extent of the data entered by the user, only the size of the editing window is limited.

The **HeightScale** property defines measurement units for the **MaxHeight** and **MinHeight** properties

### Syntax

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

```
[TListCellDef].EditInfo.TextBox.MinHeight [= minimum&]  
[TListCellDef].EditInfo.TextBox.MaxHeight [= maximum&]  
[TListCellDef].EditInfo.TextBox.HeightScale [= enum%]
```

### Values:

The **HeightScale** property settings are:

Constant	Setting	Description
TLTEXTBOX_SCALE_PIXELS	0	(Default) - <b>MaxHeight</b> and <b>MinHeight</b> properties are measured in pixels.
TLTEXTBOX_SCALE_LINES	1	<b>MaxHeight</b> and <b>MinHeight</b> properties are measured in lines of text.

The **MinHeight** property settings are:

Setting	Description
-1	(Default) – The minimum size of the editing window during editing is defined by the size of the cell.
> 0	The minimum size of of the editing window is explicitly specified in pixels or lines.

The **MaxHeight** property settings are:

Setting	Description
-1	(Default) – The maximum size of the Edit window during editing is limited by the height of the TList window.
> 0	The maximum size of of the editing window is explicitly specified in pixels or lines of text.

### Data Type

Long

### Remarks

Regardless of the **MinHeight** and **MaxHeight** properties - the size of the text box during editing is Never reduced to less that that necessary to hold the text as it exists before editing begins in an item cell.

Further control over the appearance and behavior of the text-editing window ( for instance display of scrollbars) is provided by the **Options** property of the **TextBox** object, as well as the **Options** parameter of the **RequestEditing** or **GridCellRequestEditing** event.

---

## MinWidth and MaxWidth Properties (TListTextBox Object)

### Applies to

TListTextBox object

### Description

These properties set the maximum and minimum size of the TextBox window in pixels.

### Syntax

```
[TListCellDef].EditInfo.TextBox.MinWidth [= minimum&]  
[TListCellDef].EditInfo.TextBox.MaxWidth [= maximum&]
```

### Values:

The **MinWidth** property settings are:

Setting	Description
-1	(Default) – the minimum width of the editing window during editing is defined by the size of the cell plus the additional width that is equal to two characters.
> 0	The minimum width of the editing window is explicitly specified in pixels.

The **MaxWidth** property settings are:

Setting	Description
-1	(Default) – the maximum width of the editing window during editing is defined by the right edge of the TList control window.
> 0	The maximum width of the editing window is explicitly specified in pixels.

### Data Type

Long

### Remarks

These properties may be used together with the TLTTEXTBOX\_OPT\_AUTOWIDTH flag of the **Options** (TListTextBox object) property

Regardless of the MinWidth and MaxWidth properties - the size of the text box during editing is Never reduced to less than that necessary to hold the text as it exists before editing begins in an item cell.

Further control over the appearance and behavior of the text-editing window ( for instance display of scrollbars) is provided by the Options parameter of the **RequestEditing** or **GridCellRequestEditing** events.

---

## Modifications Property

### Applies to

TList object

### Description

The Modifications property enables or disables recently added TList features to provide backwards compatibility with older editions.

This is a BitFlag property. Values for setting this property are built by OR'ing the desired bit value constants:

Write only at run-time, not available at design time.

## Syntax

[form.]TList.Modifications[= BitFlags&]

## Settings

The **Modifications** property accepts now following flags:

Setting	Value	Description
TL_MOD_CLICK_BEHAVIOR	&H01	Setting this flag sets the behavior of TList when the user clicks on the last partly visible item in tree.
TL_MOD_CELLDEF_ALIGNMENT	&H02	Setting this flag forces TList to take into consideration all parameters that could affect the height of the item when TList calculates the height of inner cell of every tree item.
TL_MOD_LISTINDEX_BEHAVIOR	&H04	Setting this flag forces TList to expand all parents (up to the root level) of an item when setting the ListIndex property.
TL_MOD_EXPAND_EVENTS	&H08	Setting this flag forces TList to generate multiple <b>Expand/Collapse</b> events in case multiple items are expanded/collapsed at once
TL_MOD_SORT_LEVELS	&H10	Setting this flag forces TList to apply sorting settings for new items according to the corresponding LevelDef for that item.
TL_MOD_IMAGE_ALIGNMENT	&H20	Setting this flag forces TList to place the item's image right after the plus/minus sign with the standard space between them.
TL_MOD_INDEX_MINUS_2	&H40	Setting this flag forces TList to change the behavior in case of using -2 as an index for properties.
TL_MOD_TLIST6	&H80	Setting this flag will cause some slight changes in TList 6 behavior (from TList 4 and 5 versions).
TL_MOD_CLICK_SELECTION	&H100	Setting this flag instructs TList not to change the ListIndex and not to select the last item in the tree when user clicks within the empty space below tree.
TL_MOD_NEW_EDITING	&H200	Setting this flag will cause some changes in in-place editing behavior. <b>We highly recommend setting this flag.</b>
TL_MOD_ROW_CELLDEF	&H400	Setting this flag will enable applying [TListGrid].RowDef(0).CellDef object's settings for grid captions.
TL_MOD_FORCE_RTF	&H800	Setting this flag will fix the problem with displaying tooltips for single-line RTF text.
TL_MOD_STRICT_TOP_INDEX	&H1000	Setting this flag will change scrolling behavior
TL_MOD_ALWAYS_USE_INVSTYLE	&H2000	Setting this flag makes possible to apply the InvStyle setting within the "tree" column of a "tree" grid object.



TL_MOD_GRID_EDITING_ADJUSTMENT	&H4000	Setting this flag bit fixes a few in-place editing related problems.
TL_MOD_IGNORE_SELECTABLE_PROPERTY	&H8000	Setting this flag forces TList to ignore any settings for Selectable property.
TL_MOD_FAST_TRANSPARENT_BITMAP	&H10000	Setting this flag forces TList to use fast method of drawing transparent the bitmaps.
TL_MOD_EXTEND_EMPTY_ITEMS	&H20000	Setting this flag forces TList to display the rows in the grid with the default height that depends on the current font settings as if there were a character there.

---

Note: Additional flags may be added in the future

---

### Default settings

Flag	Status
TL_MOD_CLICK_BEHAVIOR	ON
TL_MOD_CELLDEF_ALIGNMENT	OFF
TL_MOD_LISTINDEX_BEHAVIOR	ON
TL_MOD_EXPAND_EVENTS	ON
TL_MOD_SORT_LEVELS	OFF
TL_MOD_IMAGE_ALIGNMENT	ON
TL_MOD_INDEX_MINUS_2	ON
TL_MOD_TLIST6	ON
TL_MOD_NEW_EDITING	ON
TL_MOD_ROW_CELLDEF	ON
TL_MOD_FORCE_RTF	OFF
TL_MOD_STRICT_TOP_INDEX	OFF
TL_MOD_ALWAYS_USE_INVSTYLE	OFF
TL_MOD_GRID_EDITING_ADJUSTMENT	OFF
TL_MOD_IGNORE_SELECTABLE_PROPERTY	OFF
TL_MOD_FAST_TRANSPARENT_BITMAP	ON
TL_MOD_EXTEND_EMPTY_ITEMS	ON

### Remarks

Here is a more detailed description of all flags:

#### **TL\_MOD\_CLICK\_BEHAVIOR** flag:

Sets the behavior of TList when the user clicks on, the last partly visible item in tree.

When a user clicks on the last partly visible item in a tree, the following events are triggered:

- user press mouse button down over the partly visible item.
- TList generatesMouseDown event
- TList scrolls list up in order to make partly visible item - entirely visible
- user releases mouse button but it happens over the next item of the tree (after scrolling up)
- all following events (Click, ItemClick, GridCellClick, PictureClick...) are generated.

With this flag SET, ListIndex and the parameters of the various click events will be set to the item originally clicked even if TList scrolling causes the mouse to be over a different item on the mouse Up event.

With this flag NOT SET, ListIndex and the parameters of the ItemClick, GridCellClick, etc events reflect the item the mouse is over during the MouseUp (which may not be the same as user clicks on due to scrolling up).

**TL\_MOD\_CELLDEF\_ALIGNMENT** flag:

Setting this flag forces TList to take into consideration all parameters (ItemHeight, all item's pictures, grid row height) that could affect to the height of the item when TList calculates the height of inner cell of every tree item. The item's text can then be aligned using the real height of this item (using the ItemCell().TextAlignment property).

**TL\_MOD\_LISTINDEX\_BEHAVIOR** flag:

This flag forces TList to expand all parents (up to the root level) of an item when setting the ListIndex property. Without this flag set, setting the ListIndex to an item whose parent is collapsed has no effect.

**TL\_MOD\_EXPAND\_EVENTS** flag:

Setting this flag forces TList to generate multiple **Expand/Collapse** events in case of multiple items are expanded/collapsed at once when users use **ExpandEx** or **ExpandToLevel** properties.

---

Note: Events are generated in the top to bottom direction so events for parent will be fired before events for their children. If user tries to expand an item whose parents were also closed then **Expand** events for all item's parents will be fired before **Expand** event for the item itself and all children.

---

**TL\_MOD\_SORT\_LEVELS** flag

Setting this flag forces TList to apply sorting settings specified by corresponding LevelDef on the items immediately as they are added by different TList's properties and methods (AddItem, InsertItem, Add, LoadAndAdd, LoadAndInsert etc). Thus all LevelDef sorting settings (SortingKey, SortingStyle, SortingMethod) will be applied for new items at the corresponding level in the tree.

For optimal performance this flag should be left OFF when adding many items or otherwise modifying the tree structure, and then reset LevelDef( level).Sorted as desired. This is the default behavior.

---

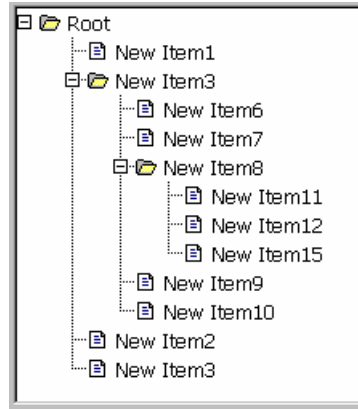
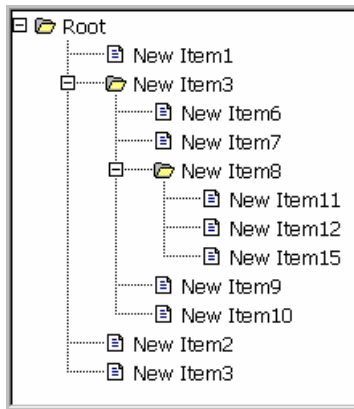
Note: While this flag is OFF, all items added after changing LevelDef sorting settings will not be sorted in a proper way. In this case the corresponding level sorting should be resorted by setting LevelDef.Sorted = True after adding new items.

---

**TL\_MOD\_IMAGE\_ALIGNMENT**

Setting this flag forces TList to place the item's image right after the plus/minus sign with the standard space between them. If this flag is not specified the position of item's image depends on ShiftStep property.

The following two pictures show the difference in the tree appearance with and without this flag (on left picture this flag is not set, on the right one is set):



## TL\_MOD\_INDEX\_MINUS\_2

Setting this flag forces TList to change the behavior in case of using -2 as an index for properties. The standard behavior of both TList 4 & 5 is that the CurrentParent property may automatically change when an Index of -2 is specified as a parameter for any property. Setting this flag in the Modifications property instructs TList NOT to change the CurrentParent value in this case. Thus the default behavior of TList has not been changed and the behavior only changes if the Modification property is set (which is not the default).

## TL\_MOD\_TLIST6

Settings this flag will cause some slight changes in TList 6 behavior (from previous TList 4,5 versions).

- 1) ItemSorted(-1) property will be reset to 0 after TList1.Clear() call.
- 2) If this flag is set TList will include data from all columns when copying from a TList Grid or ItemGrid to the clipboard in text format. If this flag is not set, then only the text from the first column will be copied in text format as per older builds.

TList always includes all data from all columns in the "TList format" on the clipboard.

## TL\_MOD\_CLICK\_SELECTION

Settings this flag instructs TList not to change the ListIndex and not to select the last item in the tree when user clicks within the empty space below tree.

## TL\_MOD\_NEW\_EDITING

- 1) For automatic editing modes (when you use **EditingMode** or **Editable** properties to enable automatic editing) TList will generate EditingKeyPress and GridCellEditingKeyPress with the corresponding key code when an end user completes the editing by pressing "Enter" key.
- 2) You can start editing other cells/items from within GridCellAfterEditing and AfterEditing event's handlers. Also you can force user to continue editing the current cell/item if it is necessary (for example because of invalid input data).
- 3) When TListTextBox object is used for editing the text, the alignment is inherited from the corresponding CellDef.TextAlignment property (in the past the alignment was always Left justified regardless of the property settings).
- 4) When there is a TreeGrid and the editing is being performed for the item in the tree column (usually first after row titles) the textbox edit object is extended all the way to right (it will take the space from the beginning of the item's text to the next column separator). Note: TLTEXTBOX\_OPT\_FULLRECT should be specified for Options property of TListTextBox object or Options parameter of GridCellRequestEditing event.
- 5) When in-place editing is started via ItemEditText or CellEdit properties and end-user presses Esc button, the editing will be cancelled as if it is the automatic editing mode.

## TL\_MOD\_ROW\_CELLDEF

Settings this flag will instructs TList to enable applying [TListGrid].RowDefs(0).CellDef object's settings for grid captions.

### **TL\_MOD\_FORCE\_RTF**

Settings this flag will fix the problem with displaying tooltips for single-line RTF text.

### **TL\_MOD\_STRICT\_TOP\_INDEX**

There was a problem in previous version of TList and when user scrolled page by page down with using vertical scrollbar or keyboard in some cases at the end of the tree TList showed several of the last items at the top of TList window and big blank space below.

In order to fix the problems DO NOT SET the flag. By default the flag is OFF.

When the flag is set, TList works as in previous builds,

### **TL\_MOD\_ALWAYS\_USE\_INVSTYLE**

If this flag is SET then InvStyle property behavior specifying display of selected cells is applied to selected cells within "tree" column of so-called "tree" grid object.

If this flag is NOT SET ( default ) then the entire grid cell in "tree" column of so-called "tree" grid object is shown as selected when gets selected by end-user or program code.

### **TL\_MOD\_GRID\_EDITING\_ADJUSTMENT**

Setting this flag bit within the Modifications property fixes following problems:

1) In-place editing of the "tree" column generated "Item" editingevents instead "Grid Cell" editing events

(for example: RequestEditing event was generated instead of GridCellRequestEditing one).

2) The TLTEXTBOX\_OPT\_FULLRECT flag for TListTextbox.Options property did not have effect for "tree" column's cells in grid.

3) A number of small problems connected with size of editing controls were fixed:

a) the width of editing box for grid cells was one pixel less than it supposed to be;

b) the height of editing box for empty "tree" grid cells was bigger that the height of editing box for non-empty "tree"

grid cells. This problem took place only when Modification flag

TL\_MOD\_CELLDEF\_ALIGNMENT is ON.

c) some other minor problems with the sizing of editing controls.

### **TL\_MOD\_IGNORE\_SELECTABLE\_PROP**

Setting this flag forces TList to ignore any settings for Selectable property.

Although user still can set/get value to Selectable property, such settings won't make any effect on displaying or on retrieving the selection related information via other properties.

Specifying this flag will speed up the operations like the one that retrieve the number of selected items/rows and some other operations.

### **TL\_MOD\_FAST\_TRANSPARENT\_BITMAP**

Settings this flag forces TList to use fast method of drawing the transparent bitmaps.

Although under some circumstances (for example when OCX used in NET applications) this method does not work for all bitmaps, so user can turn it off and return to regular slow method of drawing.

### **TL\_MOD\_EXTEND\_EMPTY\_ITEMS**

Settings this flag forces TList to display the empty rows in the grid with the default height that depends on the current font settings as if there were a character there.

---

## **MoveItem Method**

### **Applies to**

340 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX

## TList object

### Description

This method moves an item to the specified position in the tree.

### Syntax

```
TList.MoveItem(ByVal SrcIndex As Long, ByVal DestIndex As Long, ByVal Relationship As Integer) As Long
```

### Settings

The settings for Relationship parameter are:

Constant	Value	Description
TLNODERELATION_FIRST	0	First. The Node is placed before all other nodes at the same level of the node named in DestIndex parameter. In other words it becomes the first child of the same parent as the DestIndex node.
TLNODERELATION_LAST	1	Last. The Node is placed after all other nodes at the same level of the node named in DestIndex parameter.
TLNODERELATION_NEXT	2	Next. The Node is placed immediately after the node named in DestIndex parameter.
TLNODERELATION_PREVIOUS	3	Previous. The Node is placed before the node named in DestIndex parameter
TLNODERELATION_CHILD	4	Child. The Node is moved to become the last child node of the node named in DestIndex parameter.

### Return Value

This method returns the new index of the moved item since this index can differ from one specified by DestIndex parameter.

### Remarks

This method moves an item (and its children) within the tree without making any copy operations so all item's bookmarks remain valid after calling this method.

This method can't be used for moving virtual children (items whose parent has ItemVirtualParent set TRUE). But you can move the virtual parent itself around the tree. Also this method can't move an item if that move results in making it a child of virtual parent item. So you can't move virtual children items nor add new items to the existing virtual parent via this method.

### Example

```
1) To Move an item up one within its list of siblings (among children of the same parent)
DestIndex = TList.ItemPrevSibling(index)
TList.MoveItem( itemindex , DestIndex , TLNODERELATION_PREVIOUS)
```

```
2) Starting with a tree such as:
Item1 - initial index 0
    Item 1.A - initial index 1
        Item 1A.i - initial index 2
    Item 1.B - initial index 3
    Item 1.C - initial index 4
        Item 1.C.i - initial index 5
    Item 1.D - initial index 6
Item 2 - initial index 7
Item 3 - initial index 8
```

2a) Moving an Item to become child of some item  
TList.MoveItem(4 , 1, TLNODERELATION\_CHILD)

results in:

Item 1 - initial index 0  
Item 1.A - initial index 1  
Item 1A.i - initial index 2  
Item 1.C - initial index 4  
Item 1.C.i - initial index 5  
Item 1.B - initial index 3  
Item 1.D - initial index 6  
Item 2 - initial index 7  
Item 3 - initial index 8

2b) Moving an item to become first child if some item  
TList.MoveItem( 4 , 1 , TLNODERELATION\_FIRST)

results in

Item 1 - initial index 0  
Item 1.C - initial index 4  
Item 1.C.i - initial index 5  
Item 1.A - initial index 1  
Item 1A.i - initial index 2  
Item 1.B - initial index 3  
Item 1.D - initial index 6  
Item 2 - initial index 7  
Item 3 - initial index 8

---

## MoveTo Method

### Applies To

TListColDef Object

### Description

This method changes a column's position. The order in which columns are displayed is specified by a position of the **TListColDef** object in **TListColDefs** object collection.

### Syntax

TListColDefObject.MoveTo (ByVal NewPosition As Long)

### Parameters

The **MoveTo** method syntax has these parts:

Part	Description
<i>NewPosition</i>	Required. A Long representing the new position within the <b>TListColDefs</b> object collection. For the first col, <i>NewPosition</i> = <b>0</b> . To move the column to the end of the list set <i>NewPosition</i> to <b>-1</b> .

---

## MouseCol and MouseRow Properties

### Applies To

TListGrid object

### Description

The properties return the current mouse position, in row and column coordinates. Read-only.

### Syntax

```
TListGridObject.MouseCol  
TListGridObject.MouseRow  
[form.]TList.Grid.MouseCol  
[form.]TList.Grid.MouseRow  
[form.]TList.ItemGrid(ItemIndex).MouseCol  
[form.]TList.ItemGrid(ItemIndex).MouseRow
```

### Remarks

You can use these properties in code to determine where the mouse is. These properties are especially useful to display context-sensitive help on the contents of individual cells or to test whether the user has clicked on a row or column.

---

## MouseDown and MouseUp Events

### Description

These events occur when the user presses (**MouseDown**) or releases (**MouseUp**) a mouse button over a TList control.

### Syntax

```
Sub TList_MouseDown([Index As Integer], Button As Integer,  
    ➡ Shift As Integer, X As Single, Y As Single)  
Sub TList_MouseUp([Index As Integer], Button As Integer,  
    ➡ Shift As Integer, X As Single, Y As Single)
```

### Remarks

For more information, see the description of the **MouseDown** and **MouseUp** events in the *Microsoft Visual Basic Language Reference*.

### See Also

**GetItemByXY** method

---

## MouseMove Event

### Applies to

#### TList object

### Description

This event occurs when the user moves the cursor over a TList control.

### Syntax

```
Sub TList_MouseMove([Index As Integer], Button As Integer,  
    ➡ Shift As Integer, X As Single, Y As Single)
```

### Remarks

For more information, see the description of the **MouseMove** event in the *Microsoft Visual Basic On-Line Help*.

### See Also

**GetItemByXY** method

---

## MousePointer Property

### Applies to

#### TList object

### Description

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

This property determines the mouse pointer that is displayed when over the control.

### Syntax

```
[form.]TList.MousePointer[ = numeric_expr ]
```

### Remarks

For more information, see the description of the **MousePointer** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Integer

---

## Mouselcon Property

### Applies to

#### TList object

### Description

This property determines the mouse pointer that is displayed when over the control. It can be either icon or cursor. Setting this property supersede over MousePointer property.

### Syntax

```
[form.]TList.Mouselcon [= icon]
```

### Data Type

Variant

---

## MouseWheel Event

### Applies to

#### TList object

### Description

This event is triggered when user rotates mouse wheel and TList default IntelliMouse support is turned off (ie: if **WheelScrolling** property is set to TL\_WHEELSCRL\_NONE ). The event is generated only if TList is an active window. The developer can handle wheel scrolling in the desired way if he/she is not satisfied with default handling.

### Syntax

```
Sub TList1_MouseWheel( [Index As Integer,] ByVal Direction As Integer,  
    ➡ ByVal Button As Integer, ByVal Shift As Integer, ByVal XCoord As Single, ByVal YCoord  
    ➡ As Single)
```

### Parameters

Parameter	Description
Direction	Contains a mouse device dependent number that indicates the direction the mouse wheel is rotated. A negative value indicates the mouse wheel is rotated backwards; a positive value indicates the mouse wheel is rotated forwards.
Button	Contains a number that specifies which button was pressed to trigger the event: 1 (left), 2 (right), or 4 (middle).
Shift	Contains a number specifying the state of modifier keys when the mouse is pressed. The valid modifier keys are the SHIFT (1), CTRL (2), and ALT (4) keys.



XCoord, Ycoord	Contains the current horizontal (nXCoord) and vertical (nYCoord) position of the mouse pointer within the form. These coordinates are expressed in twips (VB) or in pixels (VC).
----------------	--

---

## Move Method

[Applies to](#)

[TList object](#)

[Description](#)

This method moves and/or resizes the TList control.

[Syntax](#)

```
TList.Move left[, top[, width[, height] ] ]
```

[Remarks](#)

For more information, see the description of the **Move** method in the *Microsoft Visual Basic Language Reference*.

---

## MSOutlineAdd Property

[Applies to](#)

[TList object](#)

[Description](#)

The setting of **MSOutlineAdd** determines the behavior of the **AddItem** method.

Set to True, TList implements the **AddItem** method in a method similar to the MSOutline control included with Visual Basic:

When set to False (default), TList implements the **AddItem** method as defined in the **AddItem** method.

[Syntax](#)

```
[form.]TList.MSOutlineAdd[= bool%]
```

[Data Type](#)

Boolean

[See Also](#)

The **AddItem** method for further information

---

## MS TreeView Standard Property/Method/Event

You are trying to get help on a standard MS TreeView property, method or event.

TListTreeView is compatible with MSTreeView at the property/method/event level so please see your Visual Studio documentation in order to get help on the corresponding topics.

---

## MultiLine Property

[Applies To](#)

TListCellDef Object

[Description](#)

This property specifies whether the object's data will word wrap based on the width of the column header.

### Syntax

```
TLstCellDefObject.MultiLine [ = enum%]  
[form.]TList.ItemCell(ItemIndex&).MultiLine [ = enum%]
```

### Settings

The **MultiLine** property settings are:

Name	Value	Description
TL_MULTILINE_DEFAULT	0	Use the control's <b>DefMultiLine</b> property value.
TL_MULTILINE_OLD_TRUE	-1	Word wrapping is enabled.
TL_MULTILINE_TRUE	1	Word wrapping is enabled.
TL_MULTILINE_FALSE	2	Word wrapping is disabled.

## MultiSelect Property

### Applies to

### TList object

### Description

The property determines whether a user can make multiple selections in a list box and how the multiple selections can be made.

### Syntax

```
[form.]TList.MultiSelect = enum
```

### Settings

The **MultiSelect** property settings are:

Setting	Value	Description
TLMULSEL_NONE	0	(Default)Multiple selections are not allowed.
TLMULSEL_SIMPLE	1	Simple multiple selection. A click or the Spacebar selects or deselects an item in the list.
TLMULSEL_EXT	2	Extended multiple selection. Shift+click or Shift+arrow key extends the selection from the previously selected item to the current directory. Ctrl+click selects or deselects an item in the list.
TLMULSEL_EXPLORER_LIKE	3	Windows Explorer style multiple selection support. The selection is changed on MouseDown for all situations except the case when user holds CTRL key while clicking the mouse or when clicking over an already selected item.

When the MultiSelect property is set to TLMULSEL\_EXPLORER\_LIKE TList responds as noted below:

Action	TList response
Click (no key pressed)	select item on MouseDown
click (CTRL key is pressed)	select nonselected item on MouseDown, deselect previously selected item on MouseUp
click (SHIFT key is pressed)	select a range of items on MouseDown

click (CTRL key is pressed) over non selected item + starting drag/drop	select the item that was clicked in order to include that item into a range of drag/drop items
---	--

---

**Note:** The SmartDragDrop property will not be set internally to TRUE (as it was before) when drag/drop starts. Previous to this change an item was always selected or deselected on Mouse Down regardless of the what keyboard keys were depressed.

---

### Data Type

Integer

---

## Name Property

### Applies to

TList object

### Description

This property specifies the name that must be used in code to refer to the TList control. Not available at run time.

### Remarks

For more information, see the description of the **Name** property in the *Microsoft Visual Basic On-Line Help*.

---

## NewIndex Property

### Applies to

TList object

### Description

The **NewIndex** property returns the index of item most recently added, or whose indent or shift property was most recently changed. If a number of items were added (for example using the **Add** property) the **NewIndex** property will return the index of the first added item. Not available at design time, read only at run time.

### Syntax

[form.]TList.NewIndex
-----------------------

### Remarks

For a sorted list, use of the **NewIndex** property is the only way to determine the index of newly added items.

Note that a special Index value of -3 may also be used to refer to the most recently added item, as in:

TList1.ItemFontName(-3) = "Times New Roman"
---

This is in fact faster than using the **NewIndex** property.

If CurrentIndexMethod is set to TLSYS\_LEVEL one then calling **NewIndex** property will cause a change of **CurrentParent** and **CurrentItem** properties correspondingly to the **NewIndex** parent.

### Example

TList1.AddItem "August the 15th" TList1.ItemFontName(TList1.NewIndex) = "Times New Roman"
--

### Data Type

Long

---

## Next and Prev Properties (TListNode Object)

### Applies to

TListNode object

### Description

Returns a reference to the corresponding next/previous sibling node.  
This property is read-only.

### Syntax

```
Set TListNode = Node.Next  
Set TListNode = Node.Prev
```

### Remarks

The **Prev** and **Next** properties return a reference to another **TListNode** object.

### Data Type

TListNode

### See Also

**FirstSibling** and **LastSibling** properties

---

## Node Property

### Applies to

TListNode object

### Description

This property returns a **TListNode** object that corresponds to an item specified by its index.

### Syntax

```
Set TListNode = TList.Node(ByVal ItemIndex As Long)
```

### Example

```
Dim CurNode As TListNode  
Set CurNode = TList1.Node(TList1.ListIndex)  
CurNode.SortingStyle = TL_B_SORT_IGNORE_CASE  
CurNode.SortingMethod = 0 'sort in the ascending order  
CurNode.Sorted = True 'start sorting
```

---

## NodeFromItem, NodeFromGridCell, NodeFromTListNode Methods (TListView object)

### Description

These methods returns a reference to the corresponding TreeView Node object or or an empty object (NULL) if there is no corresponding TreeView node (for instance if GridCell is TreeGrid Column Title cell).

These methods simplify converting a TList ItemIndex, TListNode or TListGridCell reference to the corresponding TreeView **Node** object.

### Syntax

```
Set [TListNode] = TreeView1.NodeFromItem(ByVal TListItemIndex As Long) As Node  
Set [TListNode] = TreeView1.NodeFromGridCell(ByVal GCellObj As TListGridCell) As Node  
Set [TListNode] = TreeView1.NodeFromTListNode(ByVal TLNodeObj As TListNode) As Node
```

## Example

Here is a sample illustrating how to use one of these methods:

```
'getting reference to Node object using reference to TListGridCell object
Sub TreeView1_TList_GridCellClick(ByVal GridCell As TListGridCell, ByVal Button As Integer)
    Dim NodeObj As Node
    Set NodeObj = TreeView1.NodeFromGridCell( GridCell )
    Debug.Print "TreeView Node key = " & NodeObj.Key
End Sub
```

---

## NoIntegralHeight Property

### Applies to

TList object

### Description

This property is not supported anymore and does nothing when accessed.

---

## NonScrollableColumns Property

### Applies to

TListGrid object (TreeGrid only, not supported for ItemGrids)

### Description

Specifies the number of columns on the left that are frozen ( can not be scrolled ) in TList. These columns will remain on left side regardless of user scrolling by way of scrollbar or keyboard. When set to 0, all columns become scrollable (even RowTitles column).

### Syntax

```
TList.Grid.NonScrollableColumns = integer
```

### Example

```
TList1.Grid.NonScrollableColumns = 4
```

### Remarks

Note: default value = 1 (only the RowTitles column is non-scrollable)

The value of the NonScrollableColumns property is based on enumeration of all columns - even hidden columns.

Thus in the case where there is no RowTitles Column, and Coldefs(2).Visible = false, setting NonScrollableColumns = 4 would mean that the 4 frozen columns are :

- Column 0 ( hidden RowTitles column )
- Column 1,
- Column 2 ( hidden )
- and Column 3

So the user would see columns 1 and 3 frozen ( not scrollable ), columns 0 and 2 will be hidden, and all other columns (columns 4, 5, 6, 7 ...) would be scrollable.

---

## NoPictureRoot Property

### Applies to

TList object

### Description

If **NoPictureRoot** is set to True, the **PictureRoot** property is ignored and:

- The **PictureLeaf** property is used as the default image for items having zero indentation and no subordinate items
- The **PictureOpen** property is used as the default image for items having zero indentation and at least one subordinate item.

Otherwise, the **PictureRoot** property specifies the default image in each of these cases.

### Syntax

```
[form.]TList.NoPictureRoot [= bool%]
```

### Data Type

Boolean

---

## OldSelStart And OldSelLength Properties (TListEditingChangeInfo object)

### Applies to

**TListEditingChangeInfo** object

### Description

These properties of the **TListEditingChangeInfo** object represent the cursor location and selection length prior to the most recent user modification during text editing within a **TListTextBox**, **TListComboBox** editing area, or **TListSpin** object.

The **OldSelStart** property returns the position of the cursor, or the starting point of selected text. The **OldSelLength** property returns the number of characters selected.

These properties are available only through **TListEditingChangeInfo** object inside **GridCellEditingChange** and **ItemEditingChange** events and should be used together with the **OldValue** property if it is necessary to get the information regarding the selection that existed just before an end-user performs an action and corresponding event (**GridCellEditingChange** and **ItemEditingChange**) was generated. This allows you to perform a rollback operation if it is necessary to disable entering some information by end-user.

### Syntax

```
[Long] = [TListEditingChangeInfo].OldSelStart  
[Long] = [TListEditingChangeInfo].OldSelLength
```

### Example

```
'DATA VERIFICATION  
Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _  
    ByVal objGridCell As TListGridCell, _  
    ByVal objChangeInfo As TListEditingChangeInfo)  
  
    ' Verify that data being entered in column 3 is a valid number  
    If GridCell.Col = 3 Then  
        'check the text and if it is not a valid number restore previous text  
        If IsNumeric(objChangeInfo.Value) = False Then  
            ObjChangeInfo.Value = objChangeInfo.OldValue 'restore data  
            ObjChangeInfo.SelStart = objChangeInfo.OldSelStart 'restore selection  
            ObjChangeInfo.SelLength = objChangeInfo.OldSelLength 'restore selection  
        End If  
    End If  
End Sub
```

### See Also

**GridCellEditingChange** and **ItemEditingChange** events, **ValueType**, **Value**, **OldValue**, **SelStart**, **SelLength**, **EditInfoObject** properties

## OLECompleteDrag event

### Applies to

TList object

### Description

This event occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.

### Syntax

```
Private Sub TList_OLECompleteDrag([effect As Long])
```

### Parameters

Parameter	Description
<i>Effect</i>	A long integer set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another). The possible values are listed in Settings.

### Settings

The settings for *Effect* parameter are:

Constant	Setting	Description
DropEffectNone	0	Drop target cannot accept the data, or the drop operation was canceled.
DropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
DropEffectMove	2	Drop results in data being moved from the drag source to the drop source. The drag source should remove the data from itself after the move.

### Remarks

The **OLECompleteDrag** event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the effect parameter of the **OLEDragDrop** event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (DropEffectMove), the source needs to delete the object from itself after the move.

If **OLEDragMode** is set to one of automatic modes, then TList handles the default behavior. The event still occurs, however, allowing the user to add to or change the behavior.

## OLEDrag method

### Applies to

TList object

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Description

This method should be called to initiate an OLE drag/drop operation.

## Syntax

```
Call [TList].OLEDrag()
```

## Remarks

When the **OLEDrag** method is called, the TList's **OLEStartDrag** event occurs, allowing it to supply data to a target component.

---

# OLEDragMode property

## Applies to

TList object

## Description

This property specifies whether TList itself or the programmer manually handles an OLE drag/drop operation. When automatic mode is ON, TList performs automatic data copying/paste operation (see **OLEDropMode** property for details).

## Syntax

```
TList.OLEDragMode [= enum%]
```

## Settings

The settings for **OLEDragMode** are:

Constant	Setting	Description
TLOLEDrag_MANUAL	0	(Default) Manual. The programmer handles all OLE drag/drop operations.
TLOLEDrag_AUTOMATIC	1	Automatic. The component handles all OLE drag/drop operations, copies data into internal buffer and performs copying operation specifies by <b>OLEDropMode</b> property.

## Data Type

Enumeration

## Remarks

When **OLEDragMode** is set to Manual, you must call the **OLEDrag** method to start dragging, which then triggers the **OLEStartDrag** event.

When **OLEDragMode** is set to **Automatic**, TList fills the **DataObject** object with the data it contains and sets the *effects* parameter before initiating the **OLEStartDrag** event (as well as the **OLESetData** and other source-level OLE drag/drop events) when the user attempts to drag out of the control. This gives you control over the drag/drop operation and allows you to intercede by adding other formats, or by overriding or disabling the automatic data and formats using the **Clear** or **SetData** methods.

If the source's **OLEDragMode** property is set to **Automatic**, and no data is loaded in the **OLEStartDrag** event, or *aftereffects* is set to **0**, then the OLE drag/drop operation does not occur.



---

**Note:** If the **DragMode** property of a control is set to **Automatic**, the setting of **OLEDragMode** is ignored, because regular Visual Basic drag and drop events take precedence over OLE Drag Drop events.

---

## OLEDragDrop Event

### Applies to

TList object

### Description

This event occurs when a source component is dropped onto TList when the source component determines that a drop can occur.

### Syntax

```
Sub TList1_OLEDragDrop([Index As Integer,]  
    data As TListDataObject,  
    effect As Long, button As Integer, shift As Integer,  
    x As Single, y As Single)
```

### Remarks

The OLEDragDrop event syntax has these parts:

Parameter	Description
index	An integer that uniquely identifies a member of a control array.
data	A <b>TListDataObject</b> object containing formats that the source will provide and possibly the data for those formats.
effect	A long integer set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
x, y	A number that specifies the current horizontal (x) and vertical (y) position of the mouse pointer within the target control. These coordinates are always expressed in terms of the target's coordinate system as set by the ScaleHeight, ScaleWidth, ScaleLeft, and ScaleTop properties.

### Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source

		should remove the data from itself after the move.
--	--	--

## OLEDropMode property

### Applies to

TList object

### Description

Specifies how a target TList component handles drop operations. When automatic mode is on TList performs automatic data copying/paste operation (see **OLEDropMode** property for details).

This property determines how source items will be dropped (as child or as sibling) when using the automatic drag/drop mechanisms.

### Syntax

<i>TList.OLEDropMode</i> [= enum%]
------------------------------------

### Settings

The settings for **OLEDropMode** are:

Constant	Setting	Description
TLOLEDROPMODE_NONE	0	(Default) None. The target component does not accept OLE drops and displays the No Drop cursor.
TLOLEDROPMODE_MANUAL	1	Manual. The target component triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.
TLOLEDROPMODE_AUTO_CHILD	2	Automatic – copy data (item/cell/row) as child object of the target pointed by mouse cursor. The target component automatically accepts OLE drops if the <b>DataObject</b> object contains data in a format it recognizes (TList's format – data should be copied automatically via using <b>OLEDragMode</b> property).
TLOLEDROPMODE_AUTO_BEFORE	4	Automatic – copy data (item/cell/row) BEFORE target object pointed by mouse cursor. The target component automatically accepts OLE drops if the <b>DataObject</b> object contains data in a format it recognizes (TList's format – data should be copied automatically via using <b>OLEDragMode</b> property).
TLOLEDROPMODE_AUTO_AFTER	8	Automatic – copy data (item/cell/row) AFTER target object pointed by mouse cursor. The target component automatically accepts OLE drops if the <b>DataObject</b> object contains data in a format it recognizes (TList's format – data should be copied automatically via using <b>OLEDragMode</b> property).

### Data Type

Enumeration

### Remarks

In order to use automatic OLE drag/drop functionality both `OLEDragMode` and `OLEDropMode` properties should be used for data source and data target `TList` components respectively.

---

## OLEDragOver Event

### Applies to

`TList` object

### Description

The event occurs when a component is dragged over `TList`.

### Syntax

```
Sub TList1_OLEDragOver(Index As Integer,
    data As TListDataObject,
    effect As Long, button As Integer, shift As Integer,
    x As Single, y As Single
    state As Integer)
```

### Remarks

The `OLEDragOver` event syntax has these parts:

Parameter	Description
index	An integer that uniquely identifies a member of a control array.
data	A <b><code>TListDataObject</code></b> object containing formats that the source will provide and possibly the data for those formats.
effect	A long integer initially set by the source object identifying all effects it supports. This parameter must be correctly set by the target component during this event. The value of effect is determined by logically Or'ing together all active effects (as listed in Settings). The target component should check these effects and other parameters to determine which actions are appropriate for it, and then set this parameter to one of the allowable effects (as specified by the source) to specify which actions will be performed if the user drops the selection on the component. The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys. For example, if both the CTRL and ALT keys are depressed, the value of shift would be 6.
x, y	A number that specifies the current horizontal (x) and vertical (y) position of the mouse pointer within the target control. These coordinates are always expressed in terms of the target's coordinate system as set by the <code>ScaleHeight</code> , <code>ScaleWidth</code> , <code>ScaleLeft</code> , and <code>ScaleTop</code> properties.
state	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control. The possible values are listed in Settings.

### Settings

The settings for *effect* are:

Constant	Value	Description
VbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
vbDropEffectScroll	2147483648 (&H80000000)	Scrolling is occurring or about to occur in the target component. This value is used in conjunction with the other values. Use it only if you are performing your own scrolling in the target component.

The settings for *state* are:

Constant	Value	Description
vbEnter	0	Source component is being dragged within the range of a target.
vbLeave	1	Source component is being dragged out of the range of a target.
vbOver	2	Source component has moved from one position in the target to another.

---

## OLEGiveFeedback event

### Applies to

TList object

### Description

Occurs after every **OLEDragOver** event. **OLEGiveFeedback** allows the source TList component to provide visual feedback to the user, such as changing the mouse cursor to indicate what will happen if the user drops the object, or provide visual feedback on the selection (in the source component) to indicate what will happen.

### Syntax

```
Private Sub TList_OLEGiveFeedback(effect As Long, defaultcursors As Boolean)
```

### Parameters

Parameter	Description
<b>Effect</b>	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Settings.
<b>DefaultCursors</b>	A boolean value which determines whether Visual Basic uses the default mouse cursor provided by the component, or uses a user-defined mouse cursor. True (default) = use default mouse cursor. False = Use cursor set by Screen.MousePointer object.

### Settings

The settings for *Effect* parameter are:

Constant	Setting	Description
----------	---------	-------------

DropEffectNone	0	Drop target cannot accept the data.
DropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation
DropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move
DropEffectScroll	(&H80000000)	Scrolling is occurring or about to occur in the target component. This value is used in conjunction with the other values. Note Use only if you are performing your own scrolling in the target component.

### Remarks

If there is no code in the OLEGiveFeedback event, or if the DefaultCursors parameter is set to True, then Visual Basic automatically sets the mouse cursor to the default cursor provided by the component.

The source component should always mask values from the effect parameter to ensure compatibility with future implementations of components. Presently, only three of the 32 bits in the effect parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, for example,

mskDropEffectCopy, such as:

If Effect = mskDropEffectCopy...

Instead, the source component should mask for the value or values being sought, as here:

If Effect And mskDropEffectCopy = mskDropEffectCopy...

-or-

If (Effect And mskDropEffectCopy)...

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

In order to use automatic OLE drag/drop functionality both OLEDragMode and OLEDropMode properties should be used for data source and data target TList components respectively.

---

## OLEStartDrag event

### Applies to

TList object

### Description

Occurs when a component's **OLEDrag** method is performed, or when a component initiates an OLE drag/drop operation when the **OLEDragMode** property is set to **Automatic**.

This event specifies the data formats and drop effects that the source component supports. It can also be used to insert data into the **DataObject** object.

### Syntax

```
Private Sub TList_OLEStartDrag(data As TListDataObject, allowedeffects As Long)
```

### Parameters

Parameter	Description
<b>Data</b>	A <b>TListDataObject</b> object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the <b>TListDataObject</b> , it is provided when the control calls the <b>GetData</b> method, and you should provide the values for the <b>data</b> parameter. The <b>SetData</b> and <b>Clear</b> methods cannot be used here.
<b>AllowedEffects</b>	A long integer containing the effects that the source component supports. The possible values are listed in Settings.

### Settings

The settings for allowedeffects parameter are:

Constant	Setting	Description
DropEffectNone	0	Drop target cannot accept the data.
DropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
DropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

### Remarks

The source component should logically OR together the supported values and place the result in the allowedeffects parameter. The target component can use this value to determine the appropriate action, and what the appropriate user feedback should be.

The **StartDrag** event also occurs if the component's **OLEDragMode** property is set to one of automatic modes. This allows you to add formats and data to the **TListDataObject** object after the component has done so. You can also override the default behavior of the component by clearing the **TListDataObject** object (using the **Clear** method) and then adding your data and formats.

You may want to defer placing data into the **TListDataObject** object until the target component requests it. This allows the source component to save time by not loading multiple data formats.

When the target performs the **GetData** method on the **TListDataObject**, the source's **OLESetData** event will occur if the requested data is not contained in the **TListDataObject**. At this point, the data can be loaded into the **TListDataObject**, which will in turn provide the data to the target.

If the user does not load any formats into the **TListDataObject**, then the drag/drop operation is canceled.

---

## OLESetData event

### Applies to

TList object

### Description

Occurs on a source component when a target component performs the **GetData** method on the source's **TListDataObject** object, but before the data for the specified format has been loaded

### Syntax

```
Private Sub TList_OLESetData (data As TListDataObject, dataformat As Integer)
```

### Parameters

Parameter	Description
<b>Data</b>	A <b>TListDataObject</b> object in which to place the requested data. The component calls the <b>SetData</b> method to load the requested format.

<b>DataFormat</b>	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the <b>TListDataObject</b> object.
-------------------	--

### Remarks

In certain cases, you may want to defer loading data into the **TListDataObject** object of a source component to save time, especially if the source component supports many formats. This event allows the source to respond to only one request for a given format of data. When this event is called, the source should check the *format* parameter to determine what needs to be loaded and then perform the **SetData** method on the **TListDataObject** object to load the data, which is then passed back to the target component.

---

## OnDragDrop and OnDragOver Methods

### Applies to

**TList** object

### Description

The **OnDragDrop** and **OnDragOver** functions must be called in the first line in the associated **DragDrop** and **DragOver** events. This is required to notify TList about these events in any application using the OCX editions where drag/drop operations are performed.

### Syntax

```
[form.]TList.OnDragDrop([index As Integer,] Source As Control, x As Single,
➡ y As Single)
[form.]TList.OnDragOver([index As Integer,] Source As Control, x As Single,
➡ y As Single, State As Integer)
```

### Remarks

The **OnDragOver** and **OnDragDrop** methods syntax have these parts:

Part	Description
index	An integer that uniquely identifies a member of a control array.
source	The control being dragged. You can refer to properties and methods with this argument, for example, Source.Visible = False.
x, y	A number that specifies the current horizontal (x) and vertical (y) position of the mouse pointer within the target control. These coordinates are always expressed in terms of the target's coordinate system as set by the ScaleHeight, ScaleWidth, ScaleLeft, and ScaleTop properties.
state	An integer that corresponds to the transition state of the control being dragged in relation to a target control: 0 = Enter (source control is being dragged within the range of a target); 1 = Leave (source control is being dragged out of the range of a target); 2 = Over (source control has moved from one position in the target to another).

---

## Options Property (TListCheckbox Object)

### Applies to

**TListCellDef** object

### Description

The set of flags for controlling the behavior of the CheckBox edit object.

### Values

The **Options** property may be set to a combination (OR) of the following flag values:

Constant	Value	Description
TLCHK_OPT_SHOW_CHECKBOX_VALUE	&H8	If the flag is set, TList shows the Checkbox state value instead of cell value in the cell.
TLCHK_OPT_CHANGE_ON_PIC_CLICK	&H100 0	If this flag is set, the CheckBox state is changed by a mouse click on the CheckBox picture. This flag is set ( on ) by default.
TLCHK_OPT_CHANGE_ON_TEXT_CLICK	&H200 0	If this flag is set, the CheckBox state is changed by a mouse click on the cell text. This flag is set ( on ) by default.
TLCHK_OPT_CHANGE_ON_CELL_CLICK	&H400 0	If this flag is set, the CheckBox state is changed by a mouse click on the cell.
TLCHK_OPT_DO_NOT_INVERT_PICS	&H800 0	If this flag is set, the CheckBox picture is not inverted for a selected cell.

Note that the CheckBox editing is impossible if none of the following flags is set :

TLCHK\_OPT\_CHANGE\_ON\_PIC\_CLICK,  
TLCHK\_OPT\_CHANGE\_ON\_TEXT\_CLICK or  
TLCHK\_OPT\_CHANGE\_ON\_CELL\_CLICK

### Syntax

[TListCellDef].EditInfo.CheckBox.Options [= flags&]
---

### Data Type

Long

---

## Options Property (TListDateTime Object)

### Applies to

TListCellDef object

### Description

The set of flags for controlling the behavior of the DateTime edit object.

### Values

The **Options** property may be set to a combination (OR) of the following values:

360 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX



Constant	Value	Description
TLDATETIME_OPT_RIGHTALIGN	&H20	If this flag is set the drop down calendar is aligned to the right side of the editing window. Otherwise the calendar is aligned to the left . This flag is ignored if the TLDATETIME_OPT_CALENDAR flag is not set.
TLDATETIME_OPT_CALENDAR	&H400	If this flag is set, a drop-down button is displayed at the right of the editing window to open a calendar display for date selection. Otherwise, spin buttons for data editing are displayed.
TLDATETIME_OPT_TODAYCIRCLE	&H800	If this flag is set, a drop-down calendar will display today's date marked with a circle.

### Syntax

```
[TListCellDef].EditInfo.DateTime.Options [= enum%] [OR enum%] [OR enum%]
```

### Data Type

Integer

## Options Property (TListSpin Object)

### Applies to

TListCellDef object

### Description

The **Options** property is a bit flag property, each bit is a flags specifying the presentation and behavior of the **TListSpin** edit object.

### Values

The **Options** property may be set to a combination (OR) of the following values:

Constant	Value	Description
TLSPIN_OPT_WRAP	&H1	If this flag is set, incrementing beyond the maximum results in starting back at the minimum value, as decrementing below the minimum returns the value to the maximum. Ex 0, 1, 2, ...Max, 0, 1, 2, ...
TLSPIN_OPT_LEFTALIGN	&H8	If this flag is set, the arrow buttons are placed at the left side of the edit field.
TLSPIN_OPT_ARROWKEYS	&H20	If this flag is set, the control value is changed by pressing the Up Arrow and Down Arrow keys. This flag is set on by default.
TLSPIN_OPT_HORZ	&H40	If this flag is set, the Spin arrows point left and right instead of up and down.

TLSPIN_OPT_NOTHOUSAND S	&H80	If this flag is set, the thousand separator is not inserted between every three decimal digit of the value
----------------------------	------	--

### Syntax

[TListCellDef].EditInfo.Spin.Options [= flags&]

### Data Type

Long

---

## Options Property (TListTextBox Object)

### Applies to

TListCellDef object

### Description

The **Options** property applied to a **TListTextBox** object determines whether the text-editing window for the associated cell or cells is automatically resized as the end-user edits the text.

### Syntax

[TListCellDef].EditInfo.TextBox.Options [= enum%]

### Values

The **Options** property may be set to either 0, or a combination (OR) of the following values:

Constant	Setting	Description
TLTEXTBOX_OPT_AUTOWIDTH	&H1	The Width of the TextBox window is automatically adjusted during editing if this flag is specified.
TLTEXTBOX_OPT_AUTOHEIGHT	&H2	The Height of the TextBox window is automatically adjusted during editing if this flag is specified.
TL_REQED_UPPERCASE	&H8	Converts all characters to uppercase as they are typed.
TL_REQED_LOWERCASE	&H10	Converts all characters to lowercase as they are typed.
TL_REQED_PASSWORD	&H20	Displays all characters as an asterisk (*) as they are typed.
TL_REQED_AUTOVSCROLL	&H40	If this flag is specified, the control automatically scrolls horizontally when the caret goes past the right edge of the control. To start a new line, the user must press ENTER.
TL_REQED_AUTOHSCROLL	&H80	If this flag is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses ENTER. The position of the word-wrap is determined by the item size.
TL_REQED_STDCOLORS	&H100	Displays edited item using standard colors.
TL_REQED_BORDER	&H200	Draws border around edited item.
TL_REQED_VSCROLLER	&H400	Displays vertical scrollbar while item is

		edited.
TL_REQED_HSCROLLER	&H200 0	Displays horizontal scrollbar while item is edited.
TL_REQED_NOMENU	&H400 0	Disable showing right click menu during editing operation.
TL_REQED_FULLRECT	&H800 0	If the flag is specified, the size of edit window is set to the size the grid cell. Note: this setting only applies for items in a grid – there is no effect upon list / tree items not in a grid cell.

### Example

```
TList1.DefItemCellDef.EditInfo.Textbox.Option = _
    TLTEXTBOX_OPT_AUTOWIDTH OR TLTEXTBOX_OPT_AUTOHEIGHT
```

Regardless of the setting of the **Options** property, the size of the text editing area is limited by the **MinHeight**, **MaxHeight**, **MinWidth** and **MaxWidth** properties, and is never less than that which would be needed to hold the text that existed in the textbox before editing.

The specifications of the Options property of the textbox may in some cases be overridden by setting the Options paramater of the RequestEditing and GridCellRequestEditing events.

TList can now automatically increase the height of the text editing area in response to wordwrapping when in multi-line

editing mode - previously the height of the editing area was only increased in response to the entry of a new

line using [Ctrl]+[Enter] key combination.

To enable this feature the TLTEXTBOX\_OPT\_AUTOHEIGHT flag must be set and the TLTEXTBOX\_OPT\_AUTOHSCROLL and TLTEXTBOX\_OPT\_HSCROLLER flags must be unset in Textbox.Options property

Example: the following code can be applied to any TListCellDef object, such as a ColDef().CellDef , or Grid.Cells().CellDef, or ItemCellDef

```
With TList1.Grid.ColDefs( 3).CellDef
    .Textbox.Options = _
        TLTEXTBOX_OPT_AUTOWIDTH OR TLTEXTBOX_OPT_AUTOHEIGHT _
        Or TLTEXTBOX_OPT_STDCOLORS Or TLTEXTBOX_OPT_BORDER _
        Or TLTEXTBOX_OPT_VSCROLLER Or TLTEXTBOX_OPT_FULLRECT
    ' Set up dimensions of the editing area in Pixels Wide x Lines tall
    Dim ColWidth as Long
    colwidth = ( TList1.Grid.ColDefs(2).Width + 240 ) / Screen.TwipsPerPixelX
    .Textbox.MaxWidth = colwidth    'max width of edit window in pixels
    .Textbox.HeightScale = TLTEXTBOX_SCALE_LINES
    .Textbox.MinHeight = 3    ' increase this if desired
    .Textbox.MaxHeight = 6    ' modify this as desired
End With
```

**Note** - that here we have set both an initial minimum textbox height of 3 lines, and a maximum height of 6 lines after which the vertical scrollbar becomes active.

## Options Property (TListComboBox Object)

### Applies to

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

**TListEditInfo** object

### Description

This property presents controls the behavior of the **TListComboBox** edit object.

### Syntax

```
[TListEditInfo].EditInfo.ComboBox.Options [= enum%] [Or enum%] [Or enum%]...
```

### Values

The **Options** property may be set to a combination (OR) of the following values:

Constant	Value	Description
TLCOMBO_OPT_SHOWLIST	&H20	Turns on automatic display of the drop-down list when editing begins. This flag is ignored when <b>Style</b> = TLCOMBO_SIMPLE
TLCOMBO_OPT_EDITAREA_AUTOWIDTH	&H100	When this flag is set, the ComboBox width is automatically changed to ensure that the edited text is always visible. The width can be changed within the range defined by the <b>MinWidth</b> , <b>MaxWidth</b> properties. If these properties are not set, the width can be changed up to the limits of the TList window
TLCOMBO_OPT_CASESENSITIVITY	&H200	When this flag is set, TList compares text entered by the user in the edit area with combo box list items in a case sensitive manner.
TLCOMBO_OPT_EDITAREA_FIT_IN_CELL	&H400	When this flag is set, the width of edit area of the combobox is determined by the size of the cell.
TLCOMBO_OPT_EDITAREA_USE_CELL_VALUE	&H800	When this flag is set, the display within the text entry area is determined by the <b>CellValue</b> property instead of the <b>DisplayValue</b> property.

---

## Pages Property

### Applies To

**TListReport** object

### Description

This property returns the collection of pages, prepared for printing. The **Pages** property returns a reference to a **TListReportPages** object collection.

Not available at design-time.

### Syntax

```
TListReportObject.Pages  
[form.]TList.Report.Pages
```

### Remarks

This collection doesn't contain any items until the **PrepareForPrinting** method is called. To fill in this collection with the information you have to specify the properties of the **TListReport** object and call the **PrepareForPrinting** method. **PrepareForPrinting** generates the PreparePage events for each page where you can change this page formatting.

### Data Type

---

## Parent Property (TListNode Object)

### Applies to

TListNode object

### Description

This property gets or sets the parent of the current node. Property returns or accepts a reference to the parent node.

### Syntax

```
Set Node.Parent[ = TListNode] or  
Set [TListNode] = Node.Parent
```

### Remarks

Using this property the developer can move any node around the tree. It is better to move a node within a sub tree using the Index property.

### Example

Starting with

```
Item 0  
Item 1  
Item 1.1  
Item 1.2  
Item 2  
Item 3
```

The following VB code:

```
Dim CurNode As TListNode  
Dim RootNode As TListNode  
Set CurNode = TList1.Node(3) ' Item 1.2  
Set RootNode = TList1.Node(0) ' Item 0  
CurNode.Parent = RootNode
```

Results in:

```
Item 0  
Item 1.2  
Item 1  
Item 1.1  
Item 2  
Item 3
```

### Data Type

TListNode

### See Also

Next, Prev, FirstSibling and LastSibling properties

---

## PasteBuffer Method

### Applies to

TList object

### Description

The **PasteBuffer** method creates a *tree buffer* and fills it with information pasted from the clipboard. The return value is zero if the function is successful. Otherwise, the return value is an error code.

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

Not available at design time.

### Syntax

```
[form.]TList1.PasteBuffer (hTreeBuffer As Long) As Integer
```

### Remarks

The parameter *hTreeBuffer* is a variable of a *Long* type. This variable is filled with a pointer to the clipboard information after the call to the **PasteBuffer** function.

---

You can use any TList control to paste any *tree buffer*.

---

## Parent Property

### Applies to

TList object

### Description

This property specifies the form in which the control is located.  
Not available at design time and read-only at run time.

### Syntax

```
[form.]TList.Parent
```

### Remarks

For more information, see the description of the **Parent** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

Form

## ParentItemIndex Property

### Applies To

TListGrid object

### Description

This property returns an index of the item, which owns the grid. This property always returns -1 for TreeGrid.

Read-only, not available at design-time.

### Syntax

```
TListGridObject.ParentItemIndex  
[form.]TList.ItemGrid(ItemIndex&).ParentItemIndex  
[form.]TList.Grid.ParentItemIndex (item&)
```

### Data Type

Long

## PathSeparator Property

### Applies to

TList object

### Description

The property sets and returns the item delimiter string used when accessing the **FullPath** property. The default value is the backslash character (\).

### Syntax

```
[form.]TList.PathSeparator[=delimiter$]
```

### Data Type

String

---

## Picture and PictureSelected Properties

### Applies To

**TListCellDef** object.

### Description

These properties set and return the picture associated with a **TListCellDef** object in normal and selected states. **TListCellDef** objects are used to specify graphics for grid cells.

### Syntax

```
TListCellDefObject.Picture [=picture]  
TListCellDefObject.PictureSelected [=picture]  
[form.]TList.Grid.Cells(Ro&, Col&).CellDef.Picture [=picture]  
[form.]TList.Grid.Cells(Ro&, Col&).CellDef.PictureSelected [=picture]  
[form.]TList.LevelDef(Index&).CellDef.Picture [=picture]
```

### Example

The code below sets pictures which should be shown in a cell:

```
TList1.Grid.Cells(10, 10).CellDef.Picture = _  
    LoadPicture("Off.BMP")  
TList1.Grid.Cells(10, 10).CellDef.PictureSelected = _  
    LoadPicture("On.BMP")
```

### REMARK

Setting the editing style for a TList cell or cells to Checkbox disables the effect of the Picture and PictureSelected properties for that cell or cells.

---

## Picture... Properties

### Applies To

**Picture...** properties apply to **TList** control.

**PictureClosed**, **PictureLeaf**, **PictureOpen** properties apply also to **TListLevelDef** object.

### Description

These properties set and return the picture associated with items in particular states.

### Syntax

```
[form.]TList.PictureRoot[=picture]  
[form.]TList.PictureOpen[=picture]  
[form.]TList.PictureClosed[=picture]  
[form.]TList.PictureLeaf[=picture]  
[form.]TList.PictureInverted[=picture]  
[form.]TList.LevelDefs(IndentationLevel&).PictureClosed [=picture]  
[form.]TList.LevelDefs(IndentationLevel&).PictureLeaf[=picture]  
[form.]TList.LevelDefs(IndentationLevel&).PictureOpen[=picture]
```

### Remarks

The graphic displayed for a given item is most directly determined by the **Image** and **InvImage** properties set for that individual item. If however, either of these two properties are not set in code,

they are automatically set by TList to the appropriate **Picture...** properties as determined by the current state of the item (closed, open, root, etc) and the **PictureType** property.

The **PictureInverted** property determines the graphic referenced by the **InvImage** property (an array) corresponding to a given item unless the value of **InvImage** for that item is otherwise directly set to a non-zero value in code.

The **PictureRoot** property determines the graphic referenced by the **Image** property (an array) for an item having a corresponding **Shift** property value of 0 (such an item has no parents) unless the value of **InvImage** for that item is otherwise directly set to a non-zero value in code. This may be overridden by use of the **NoPictureRoot** property

For all non-root items the **PictureLeaf** property defines the **Image** property value for items without subordinates, **PictureOpen** defines the value when there are visible subordinates items, and **PictureClosed** defines the value for items having all subordinates hidden.

When **PictureClosed**, **PictureLeaf**, **PictureOpen** properties apply to **TListLevelDef** object they specify default item pictures for items of a given indentation level.

These properties can display either bitmap files (\*.BMP) or icon files (\*.ICO) or metafiles (\*.WMF). Setting of these properties updates the control unless the **Redraw** property is set to False.

At run time, any of these properties can be set to any other object's **DragIcon**, **Icon**, **Image**, or **Picture** property, or you can assign it the graphic returned by the **LoadPicture** method. These properties can only be assigned directly.

No picture is displayed if the control's **ViewStyle** property is set to a value greater than 3.

---

## PicturePalette Property

### Applies to

TList object

### Description

This property sets and returns the picture which palette is used to display all pictures in TList.

### Syntax

```
[form.]TList.PicturePalette[=picture]
```

### Remarks

This property can be only DIB file (\*.DIB).

---

## PicInMultiLine Property

### Applies to

TList object

### Description

Setting the **PicInMultiLine** property determines the vertical alignment of a picture associated with an item having multiple lines of text (see **ItemMultiLine** and **WidthOfText** properties).

### Syntax

```
[form.]TList.PicInMultiLine[ = enum%]
```

### Settings

The **PicInMultiLine** property settings are:

Setting	Description
0	(Default) Picture at top.



1	Picture in the middle.
2	Picture at the bottom

---

## PictureClick Event

### Applies to

TList object

### Description

This event is generated whenever a picture associated with an item is clicked.

### Syntax

```
Sub TList_PictureClick([Index As Integer])
```

### Remarks

To determine the item whose picture was double-clicked use the **ListIndex** property.

---

## PictureDbClick Event

### Applies to

TList object

### Description

The event is generated whenever a picture associated with an item is double-clicked.

### Syntax

```
Sub TList_PictureDbClick( [Index As Integer] )
```

### Remarks

To determine the item whose picture was double-clicked use the **ListIndex** property.

---

## PictureMark Property

### Applies to

TList object

### Description

This property specifies the default picture assigned to elements of the **MarkPicture** array.

### Syntax

```
[form.]TList.PictureMark [ = picture]
```

### Remarks

To display mark pictures, the **ViewStyleEx** property must be set to 2 or 3.

### Data Type

Object

### See Also

ItemMark, MarkPicture, MarkTag, MarkHeight, MarkWidth, and ViewStyleEx

---

## PicturePlus and PictureMinus Properties

### Applies to

**TList** object

### Description

The **PicturePlus** and **PictureMinus** properties are used to designate images displayed within the tree lines for items which can be expanded or collapsed.

### Syntax

```
[form.]TList.PicturePlus [= Picture]
[form.]TList.PictureMinus [= Picture]
```

### Remarks

The **PictureMinus** property defines the picture used for an item which can be collapsed to hide its currently displayed children. The **PicturePlus** property defines the picture used for an item which can be expanded to display its currently hidden children.

If **PicturePlus** / **PictureMinus** property is not set to any picture, TList draws a default plus and/or minus image resembling those used by the Windows 95 Explorer .

The result of double clicking on the Plus or Minus picture is controlled by TList's **AutoExpand** property.

To display plus/minus pictures, the **ViewStyleEx** property must be set to 1 or 2.

### See Also

**PicturePlus**, **PictureMinus**, **AutoExpand**, and **ViewStyleEx**

---

## PictureType Property

### Applies to

**TList** object

### Description

This property sets and returns the type of picture that appears for each item in the TList control.

### Syntax

```
[form.]TList.PictureType [= enum%]
```

### Settings

The **PictureType** property settings are:

Setting	Description
0	(Default) The picture that appears for each item is determined by the type of the item.
1	The picture of each item is specified by the <b>PictureRoot</b> property
2	The picture of each item is specified by the <b>PictureOpen</b> property
3	The picture of each item is specified by the <b>PictureClosed</b> property
4	The picture of each item is specified by the <b>PictureLeaf</b> property

### Remarks

Setting of the **PictureType** property updates the control unless the **Redraw** property is set to False.

### Data Type

Integer

---

## PictureWidth and PictureHeight Properties

### Applies To

370 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX

## TListCellDef object

### Description

These properties determine the width and height of the picture displayed in a cell. These properties are measured in twips.

Not available at design-time.

### Syntax

```
[form.]TList1.Grid.Cells(R, C).CellDef.PictureWidth = [twips%]  
[form.]TList1.Grid.Cells(R, C).CellDef.PictureHeight = [twips%]  
  
[form.]TList1.Grid.ColDef(C).CellDef.PictureWidth = [twips%]  
[form.]TList1.Grid.ColDef(C).CellDef.PictureHeight = [twips%]  
  
[form.]TList1.ItemGrid(ItemIndex&).ColDef(C).CellDef.PictureWidth = [twips%]  
[form.]TList1.ItemGrid(ItemIndex&).ColDef(C).CellDef.PictureHeight = [twips%]  
  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.PictureWidth = [twips%]  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.PictureHeight = [twips%]  
  
[form.]TList1.ItemCellDef(ItemIndex&).PictureWidth = [twips%]  
[form.]TList1.ItemCellDef(ItemIndex&).PictureHeight = [twips%]  
  
[form.]TList1.DefItemCellDef.PictureWidth = [twips%]  
[form.]TList1.DefItemCellDef.PictureHeight = [twips%]
```

### Settings

These properties' settings are:

Setting	Description
0	(Default) The picture in a cell is displayed using its original size.
> 0	The size of the picture is expressed in terms of the container's scale. By default measured in twips.

### Data Type

Long

---

## PlusMinusClick and PlusMinusDbClick Events

### Applies to

TList object

### Description

These events are generated whenever a plus/minus picture associated with an item is clicked or double clicked.

### Declarations

```
Sub TList_PlusMinusClick( [Index As Integer], ByVal / As Long )  
Sub TList_PlusMinusDbClick( [Index As Integer], ByVal / As Long )
```

---

## PostScriptDC Property

### Applies To

TListReport object

### Description

This property specifies that the target printer is a postscript printer.

Setting this property to True state force TList prints icons like bitmaps (without transparent regions) because postscript printers don't print icons properly.

---

**Note:** TList now automatically detects postscript printers  
The PostscriptDC property is now ignored and printer type detection is automatic.

---

### Syntax

```
TListReportObject.PostScriptDC[= boolean% ]  
[form.]TList.Report.PostScriptDC [= boolean% ]
```

### Data Type

Integer (boolean)

---

## PrepareForPrinting Method

### Applies To

TListReport Object

### Description

The **PrepareForPrinting** method is used for page repagination and creates the **Pages** collection, which holds the results of the repaginating. During this operation you can change settings for each page separately handling the **PreparePage** event. This method returns the number of pages available for printing. This method is helpful if you want to print only a few of the selected pages.

Note this method is called automatically by the **Start** method if wasn't called manually before, or if properties of the **Report** object were changed after the previous call to the **PrepareForPrinting** method.

### Syntax

```
[form.]TList.Report.PrepareForPrinting() As Long
```

---

## PreparePage Event

### Applies to

TListReport object

### Description

This event is generated for each page after the **PrepareForPrinting** method has been called. You can handle this event to control the Pages collection creation. This event is used to specify the settings for each page separately.

### Usage

```
Sub TList_PreparePage([Index As Integer,] PrinterObject As Variant,  
    ➔ ReportPage As TListReportPage)
```

### Remarks

The **PreparePage** event syntax has these parts:

Part	Description
Index	An integer that uniquely identifies a control if it is in a control array.
PrinterObject	An output device, can be a printer object, a picture-box, or a DC handle
ReportPage	An object that describes the current page, see TListReportPage properties and methods

---

## PreviewMode Property

### Applies To

TListReport object

### Description

This property specifies if TList should print on a printer or a screen DC (usually used to make a preview).

Not available at design-time.

### Syntax

```
TListReportObject.PreviewMode [ = mode%]  
[form.]TList.Report.PreviewMode [ = mode%]
```

### Remarks

If this property is set to True, TList goes in the Preview mode and the size of a page will be calculated as specified by the **PreviewPageWidth** and the **PreviewPageHeight** properties. Otherwise the size of a page will be taken from a DC of the **Printer** object as specified by the **PrinterObject** property.

### Data Type

Integer (Boolean)

---

## PreviewPageWidth and PreviewPageHeight Properties

### Applies To

TListReport object

### Description

These properties specify the width and the height of a page to be printed. These properties are measured in twips.

Not available at design-time.

### Syntax

```
TListReportObject.PreviewPageWidth [ = WidthInTwips&]  
[form.]TList.Report.PreviewPageWidth [ = WidthInTwips&]  
TListReportObject.PreviewPageHeight [ = HeightInTwips&]  
[form.]TList.Report.PreviewPageHeight [ = HeightInTwips&]
```

### Data Type

Long

---

## PrintBackground Property

### Applies To

TListReport object

### Description

This property specifies if TList should print its background.

Not available at design-time.

### Syntax

```
TListReportObject.PrintBackground[= boolean% ]  
[form.]TList.Report.PrintBackground [ = boolean% ]
```

### Data Type

Integer (boolean)

---

## PrintLevels Property

### Applies To

TListReport object

### Description

This property specifies how many levels will be printed.

Not available at design-time.

### Syntax

```
TListReportObject.PrintLevels [ = enum% ]  
[form.]TList.Report.PrintLevels [ = enum% ]
```

### Settings

The **PrintLevels** property settings are:

Setting	Description
-1	Print as displayed on the screen
0	Print only root items.
1	Print only root items and their direct children
2	Print only root items, their direct children, and direct grand-children
3	Print only root items, their direct children, direct grand-children, and direct grand-grand-children
...	...

### Data Type

Integer

---

## PrinterObject Property

### Applies To

TListReport object

### Description

This property can be set either to the **Printer** object, a **PictureBox** control, or a DC handle. TList automatically determines what exactly has been passed to this property

Not available at design-time.

### Syntax

```
TListReportObject.PrinterObject [= object ]  
[form.]TList.Report.PrinterObject [= object ]
```

### Data Type

Printer Object, PictureBox Control, Integer

---

## PrintingStop Method

### Applies To

TListReport Object

### Description

The **PrintingStop** method is used to cancel the printing process programmatically. Note that this method doesn't clear the printer spooler so you have to manually complete or abort the current printing process either calling the **EndDoc** or the **KillDoc** method.

### Syntax

```
[form.]TList.Report.PrintingStop
```

## PrintOneStep Method

### Applies to

**TList** object

### Description

This method provides simplified printing functionality. This method allows user to specify output DC, margins, header, footer and some more options for further printing. For using advanced printing functionality it is better to use Report object interface.

Not available at design time.

### Syntax

```
[form.]TList.PrintOneStep(OutputDC As Variant, LeftMargin As Long, TopMargin As Long, RightMargin As Long, BottomMargin As Long, FirstItem As Long, LastItem As Long, Options As Long, Header As String, Footer As String, [Zoom] As Variant)
```

### Remarks

The **PrintOneStep** method parameters:

Part	Description
<i>OutputDC</i>	Output DC that will be used for further printing.
<i>XXXXMargin</i>	Specify the region on a page where TList will print data. These properties are measured in twips.
<i>FirstItem</i>	First item that will be printed.
<i>LastItem</i>	Last item that will be printed.
<i>Options</i>	Printing options.
<i>Header</i>	Header string will be printed on a top of each page. Should be set to empty string for disable header printing.
<i>Footer</i>	Footer string will be printed on the bottom of each page. Should be set to empty string for disable header printing
<i>Zoom</i>	Zoom factor in percents (from 1% to 10000%).

The **OutputDC** parameter can have following values:

Value	Description
-1	TList uses default printer.
-2	Show printer dialog and print on the choosen printer.
<i>Printer object or DC</i>	Valid VB printer object or printer DC handle

Following flags can be used for **Options** parameter:

Constant	Value	Description
TL_PRNOS_COLTITLES_NONE	&H0	No TreeGrid headers will be printed.
TL_PRNOS_COLTITLES_FIRSTPAGEONLY	&H1	TreeGrid header will be printed only on first page.

TL_PRNOS_COLTITLES_ALWAYS	&H2	TreeGrid header will be printed on each page.
TL_PRNOS_FULL_EXPAND	&H4	Expand all branches before printing and restore tree state after.
TL_PRNOS_BACKGROUND	&H8	Print TList background.
TL_PRNOS_ABORT_STYLE	&H10	Do not show predefined abort window.
TL_PRNOS_GENERATE_EVENTS	&H20	Generate TList printing events PreparePage, BeginPage, EndPage.
TL_PRNOS_DO_NOT_INIT	&H40	Do not initialize output DC before printing.
TL_PRNOS_POST_SCRIPT_DC	&H80	TList prints icons like bitmaps. (Postscript printers don't print icons properly)

**Note:** Header and footer strings will be printed using TList settings (Font, ForeColor, BackColor, Multiline settings). To disable printing any of these strings it is necessary to specify empty string for corresponding parameter. Also there is a possibility to specify page number inside header or footer string. Every "<p>" substring inside header or footer will be replaced with current page number.

---

## RecalculateSize Method (TListTooltipWindow Object)

### Applies to

TListTooltipWindow object

### Description

Calling this method **AFTER** setting the Text, Picture, and Style properties, instructs TList to recalculate the required window size for the specified tooltip contents.

### Syntax

```
[TListTooltipWindow].RecalculateSize (ByVal bKeepWidth As Boolean)
```

### Settings

Calling this method with parameter TRUE will instruct TList to use the maximum possible width of the text for the item/cell under the cursor ( generally this is the screen width ).

Calling this method with parameter FALSE will instruct TList to calculate the width for the tooltip window based on the tooltip text and picture size. In either case ( True or False ) the tooltip height will be calculated to fit the text which is wrapped within the calculated width.

### Remarks

This method call will reset the height and possibly even the width of the tooltip window even if they were explicitly assigned earlier via LocalPosition and ScreenPosition properties.

### Example

```
Sub TList1_ShowTooltip(ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    .Window.LocalPosition.Left = .MouseLocal.X
    .Window.LocalPosition.Top = .MouseLocal.Y
    .Window.RecalculateSize False
  End With
End Sub
```

### See Also

TListTooltipWindow object

376 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX



---

# Redraw Property

## Applies to

TList object

## Description

This property determines whether the TList control displays updates as they are made. Updates are not displayed while the **Redraw** property is False. Upon setting **Redraw** to True, any changes, will be displayed, as will any future changes made while the property remains True. Setting **Redraw** to True doesn't always cause updating of the control. Only *changed* items will be updated.

Not available at design time.

## Syntax

```
[form.]TList.Redraw[= {True/False}]
```

## Remarks

It is generally advisable to set the **Redraw** property to False before making changes to the control which cause numerous repaint events. Then reset **Redraw** back to True when you are done. For example:

```
TList1.Redraw = False
For I% = 1 to 1000
    TList1.AddItem "Item" & I%
Next
TList1.Redraw = True
```

The **Refresh** method automatically sets the **Redraw** property to True.

It is possible to use recursive calls to the **Redraw** property:

```
TList1.Redraw = False
AddItemsProcedure
TList1.Redraw = True
...
Sub AddItemsProcedure
    TList1.Redraw = False
    For I% = 1 to 1000
        TList1.AddItem "Item" & I%
    Next
    TList1.Redraw = True
End Sub
```

TList counts the number of calls to **Redraw** property and repaints itself only when the **Redraw** property was reset to True as many times as it had been set to False .

## Data Type

Boolean

---

# Redraw Property (TListTreeView)

## Description

This property determines whether the TListTreeView control displays updates as they are made. Updates are not displayed while the **Redraw** property is False. Upon setting **Redraw** to True, any changes, will be displayed, as will any future changes made while the property remains True. Setting **Redraw** to True doesn't always cause updating of the control. Only *changed* items will be updated. Not available at design time.

## Syntax

```
[form.]TListTreeView.Redraw[= {True/False}]
```

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Remarks

It is generally advisable to set the **Redraw** property to False before making changes to the control which cause numerous repaint events. Then reset **Redraw** back to True when you are done. This will GREATLY improve performance. For example:

```
TListTreeView1.Redraw = False
For I% = 1 to 1000
    TListTreeView1.Nodes.Add , twwLast, , "Item " & I%
Next
TListTreeView1.Redraw = True
```

---

Note: There is no difference between using TListTreeView.Redraw and using TListTreeView.TList.Redraw so the developer can use them interchangeably.

---

## Data Type

Boolean

---

# Refresh Method

## Applies to

TList object

## Description

This method updates a control at run time.

## Syntax

```
TList.Refresh
```

## Remarks

For more information, see the description of the **Refresh** method in the *Microsoft Visual Basic On-Line Help*.

---

# RefreshItems Method

## Applies to

TList object

## Description

This method generates **ItemQueryData** for the specified range of virtual items.

## Syntax

```
Sub TList.RefreshItems(ByVal IndexFrom As Long, ByVal IndexTo As Long)
```

## Example

```
' This example updates virtual children of a given parent
NumChildren = TList1.ItemVirtualCount(ParentID)
If NumChildren > 0 then
    TList1.RefreshItems ParentID+1, ParentID + NumChildren
End If
```

---

# Remove Method (TListComboltems Object)

## Applies to

TListEditInfo object

## Description

This method removes an item, as specified by a unique key (**CellValue**), from the **TListComboItems** collection.

## Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.Remove CellValue,  
(Where CellValue is the value corresponding to the value of the TListComboItem object. CellValue is a first parameter  
in the Add method of the TListComboItems object.)
```

## Example

```
' add four items to the combo list with pictures  
With TListEditInfo.ComboBox.Items  
.Add 10, Picture1.Picture, "Ten"  
.Add 20, Picture2.Picture, "Twenty"  
.Add 30, Picture3.Picture, "Thirty"  
.Add 40, Picture4.Picture, "Forty"  
... some other code  
' remove item "Forty" from the combo list  
.Remove 40  
End With
```

---

# Remove Method (TListNode Object)

## Applies to

**TListNode** object

## Description

Removes a specified Node from a **TListNode** collection.

## Syntax

```
Nodes.Remove(ByVal NodeIndex As Variant)
```

## Settings

The **Remove** method has following parameter:

Parameters	Description
NodeIndex	An integer that uniquely identifies the node object within the collection.

---

# Remove Method (TListSelectedGridCells, TListSelectedGridColumn and TListSelectedGridRows Objects)

## Applies to

**TListGrid** object

## Description

This method removes object from the a collection that holds a series of **TListGridCell**, **TListColDef** and **TListRowDef** objects representing all selected cells/columns or rows. This is the same as deselecting corresponding cell,column or row visible object of the control.

## Syntax

```
[TListGrid].SelectedCells.Remove(ByVal IIndex As Long)
[TListGrid].SelectedColumns.Remove(ByVal IIndex As Long)
[TListGrid].SelectedRows.Remove(ByVal IIndex As Long)
```

## Values

The method's parameters are:

Parameter	Description
<b>IIndex</b>	An index of the object in the collection.

## Details

Attempting to remove an object using incorrect index will trigger an error.

## Example

a) Removing two cells from the collection (deselecting them)

```
'deselect two last cells of the collection
Dim objSelectedCells as TListSelectedGridCells
Set objSelectedCells = TList1.Grid.SelectedCells
Call objSelectedCells.Remove(TList1.Grid.SelectedCells.Count-1)
Call objSelectedCells.Remove(TList1.Grid.SelectedCells.Count-1)
'display results of the deselection
Dim objCell as TListGridCell
Debug.Print "Selected Cells"
For Each objCell In objSelectedCells
    Debug.Print "objCell =" & objCell.Row & "," & objCell.Col & ")"
Next
```

b) Removing two rows from the collection (deselecting them)

```
'deselect first two rows of the collection
Dim objSelectedRows as TListSelectedGridRows
Set objSelectedRows = TList1.Grid.SelectedRows
Call objSelectedRows.Add(0)
Call objSelectedRows.Add(0)
'display results of the deselection
Dim objRowDef as TListRowDef
Debug.Print "Selected Rows"
For Each objRowDef In objSelectedRows
    Debug.Print "Row Index =" & objRowDef.Index
Next
```

c) Removing two columns from the collection (deselecting them)

```
'deselect last column in the collection
Dim objSelectedColumns as TListSelectedGridColumn
Set objSelectedColumns = TList1.Grid.SelectedColumns
Call objSelectedColumns.Remove(TList1.Grid.SelectedColumns.Count-1)
'display results of the deselection
Dim objColDef as TListColDef
Debug.Print "Selected Columns"
For Each objColDef In objSelectedColumns
    Debug.Print "ColDef Index =" & objColDef.Index & objColDef.ValueName
Next
```

---

# Remove Method (TVWSelectedNodes Object)

## Description

Deselects a currently selected node.

## Syntax

```
[form.]TListView.SelectedNodes.Remove(objNode As Node)
```

## Settings

The **Remove** method has one parameter:

Parameters	Description
objNode	A reference to either the selected node object, or an index in SelectedNodes collection

## Example

```
' Deselect one node using Remove method of TVwSelectedNodes collection
TListView.SelectedNodes.Remove TListView.Nodes(SomeNodeIndex)
' Or
TListView.SelectedNodes.Remove SelectedNodeIndex

' Deselect the last Selected Node
With TListView.SelectedNodes
.Remove .item( .Count-1 )
End With
```

## Remarks

You can also deselect nodes using Selected property of the Node object

```
TListView.Nodes(Index).Selected=False.
```

This action will automatically update SelectedNodes collection.

---

# RemoveItem Method

## Applies to

TList object

## Description

The method removes an item and subordinate items from the TList control at run time.

## Syntax

```
TList.RemoveItem index&
```

## Remarks

The **RemoveItem** method removes both the specified item and all its subordinate items. Calls to this method update the control unless the **Redraw** property is set to

False. **RemoveItem Method (TListComboItems Object)**

## Applies to

TListEditInfo object

## Description

This method removes an item specified by its zero-based index from the TListComboItems collection.

## Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.RemoveItem index%
```

## Example

```
With TListEditInfo.ComboBox.Items
' add four items to the combo list with pictures
.Add 10, Picture1.Picture, "Ten"
.Add 20, Picture2.Picture, "Twenty"
```

```
.Add 30, Picture3.Picture, "Thirty"  
.Add 40, Picture4.Picture, "Forty"  
'... some other code  
' remove item "30" from the combo list  
.RemoveItem 2  
End With
```

---

## RemoveRow Method

### Applies To

TListGrid Object

### Description

This method removes a row from a Grid object. The **RemoveRow** method doesn't support named arguments.

### Syntax

```
TListGridObject.RemoveRow (ByVal index As Long)
```

### Parameters

The **RemoveRow** method syntax has these parts:

Part	Description
<i>Index</i>	Required. A Long representing the position within the control from where the row is removed. For the first row, <i>index</i> =0.

---

## Report Property

### Applies To

TList control

### Description

This property returns a reference to the **TListReport** object, which controls the printing process with TList. The **Report** property can be set only to the Visual Basic constant, **Nothing**. Otherwise it is a read-only property:

### Syntax

```
[form.]TList.Report
```

### Data Type

Object

---

## RequestEditing Event

### Applies to

TList object

### Description

This event is triggered when the **ItemEditText** property is set to 2 (BEGIN) but before any actual editing takes place.

### Syntax

```
Sub TList1_ RequestEditing ( [Index As Integer,] Cancel As Integer,  
    ➞ ByVal ItemIndex As Long, TextToEdit As String, Options As Integer)
```

## Parameters

Parameter	Description
Cancel	Initially set to False. If you set it to <u>True</u> , editing is canceled.
ItemIndex	The index of the item being edited.
TextToEdit	Initially this is the same as text of the item being edited, but you can modify this text, changing what is seen in the edit box.
Option	This parameter is a sum composed of specified flag bits (see following table), you can change any of these settings in the <b>RequestEditing</b> event procedure. <b>Note:</b> some of these flags can be ignored. This limitations depend on the type of the editor is called for editing the item (no limitations only for textbox editors). See TListEditInfo object for details.

Option parameter flags:

Constant	Value	Description
TL_REQED_LEFT	&H0	Left aligns text.
TL_REQED_CENTER	&H1	Centers text.
TL_REQED_RIGHT	&H2	Right aligns text.
TL_REQED_MULTILINE	&H4	Designates multiline editing.
TL_REQED_UPPERCASE	&H8	Converts all characters to uppercase as they are typed.
TL_REQED_LOWERCASE	&H10	Converts all characters to lowercase as they are typed..
TL_REQED_PASSWORD	&H20	Displays all characters as an asterisk (*) as they are typed.
TL_REQED_AUTOVSCROLL	&H40	If this flag is specified, the control automatically scrolls horizontally when the caret goes past the right edge of the control. To start a new line, the user must press ENTER.
TL_REQED_AUTOHSCROLL	&H80	If this flag is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses ENTER. The position of the wordwrap is determined by the item size.
TL_REQED_READONLY	&H800	Prevents the user from typing or editing text.
TL_REQED_STDCOLORS	&H100	Displays edited item using standard colors.
TL_REQED_BORDER	&H200	Draws border around edited item.
TL_REQED_VSCROLLER	&H400	Displays vertical scrollbar while item is edited.
TL_REQED_HSCROLLER	&H200 0	Displays horizontal scrollbar while item is edited.
TL_REQED_NOMENU	&H400 0	Disable showing right click menu during editing operation.

---

## Root Property

### Applies to

TListNodes object

### Description

This property returns a **TListNodes** root collection containing all items at zero level (items without indentation) .

### Syntax

```
Nodes = TList.Root
```

### Example

```
' Remove the 4th root in the tree  
' remember TList starts counting at 0  
TList.Root.Remove(3 )
```

---

## RowCellDef Property

### Applies To

TListGrid object

### Description

This property returns a reference to a **TListCellDef** object, which collects all default properties' settings (like color, font etc.) for a specified row in a grid.

Not available at design time.

### Syntax

```
[form.]TList1.Grid.RowCellDef(RowIndex&)  
[form.]TList1.ItemGrid(ItemIndex&).RowCellDef(RowIndex&)
```

### Remarks

The effect of this property includes modification of the row title cell for the specified row.

### Example

The statement below will change the background color of the 10<sup>th</sup> row in a grid:

```
TList1.Grid.RowCellDef(10).BackColor = RGB(127, 127, 127)
```

---

## RowDefs Property (TListGrid object)

### Applies to

TListGrid object

### Description

This property returns a reference to a **TListRowDefs** object. This object represents a set of rows in a grid object. See **TListRowDefs** object for details.

### Syntax

```
[TListRowDefs] = [TListGrid].RowDefs
```

---

## RowHeight Property

### Applies To

TListGrid object



## Description

This property returns or sets the height of a given row of a grid. Not available at design time.

## Syntax

```
TListGridObject.RowHeight(Row&) [= number& ]  
[form.]TList.Grid.RowHeight(Row&) [= number& ]  
[form.]TList.ItemGrid(ItemIndex&).RowHeight(Row&) [= number& ]
```

## Settings

The **RowHeight** property settings are:

Setting	Description
-1	(Default) The row height will be calculated. Row zero cannot be set to this value.
> -1	Specifies the height of a row

## Data Type

Single

## See Also

**ItemHeight** property

---

# RowTitleCellDef Property

## Applies To

**TListGrid** object

## Description

This property returns a reference to a **TListCellDef** object. This object collects all properties which determine the default formatting for row titles. This property doesn't affect formatting of either other grid cells or column titles.

## Syntax

```
[form.]TList1.Grid.ColTitleCellDef  
[form.]TList1.ItemGrid(ItemIndex&).ColTitleCellDef
```

## Example

```
TList1.Grid.ColTitleCellDef.BackColor = RGB(127, 127, 127)
```

---

# RowTitlesWidth Property

## Applies To

**TListGrid** object

## Description

This property returns or sets the width of the row titles area of a grid (the first column shown if **ShowRowTitles** is set to True). Measured in twips. Not available at design time.

## Syntax

```
TListGridObject.RowTitlesWidth [= number& ]  
[form.]TList.Grid.RowTitlesWidth [= number& ]  
[form.]TList.ItemGrid(ItemIndex&).RowTitlesWidth [= number& ]
```

## Data Type

Single

---

## RowToItemIndex Method

### Applies To

TListGrid Object

### Description

This method converts from a row number within a TList Grid object to an index value for TList. If the row index doesn't match any item, -1 is returned.

### Syntax

```
TListGridObject.RowToItemIndex(ByVal RowIndex As Long) As Long
```

### Example

```
Sub GridCellDbClick(GridCell As TListGridCell, Button As Integer)
    ' user has clicked on a grid cell,
    ' lets identify which index item this belongs to
    Index = GridCell.Grid.RowToItemIndex(GridCell.Row)
End Sub
```

---

## RTFStyle Property

### Applies To

TListCellDef object

### Description

The **RTFStyle** property determines whether TList interprets and displays text for an item or item grid cell as RTF formatted text.

### Syntax

```
[form.]TList1.Grid.Cells(R, C).CellDef.RTFStyle = [enum%]  
[form.]TList1.Grid.ColDefs(C).CellDef.RTFStyle = [enum%]  
[form.]TList1.ItemGrid(ItemIndex&).ColDefs(C).CellDef.RTFStyle = [enum%]  
[form.]TList1.ItemGrid(ItemIndex&).Cells(R, C).CellDef.RTFStyle = [enum%]  
[form.]TList1.ItemCellDef(ItemIndex&).RTFStyle = [enum%]  
[form.]TList1.DefItemCellDef.RTFStyle = [enum%]
```

### Settings

The **RTFStyle** property settings are:

Setting	Description
0	ASCII formatted text
1	RTF formatted text

### Remarks

TList supports the same RTF formatting codes, and has the same limitations, as the TextRTF property of the MS RichTextBox control.

### Data Type

Integer

---

## Save Property

### Applies to

TList object

### Description

This property saves the item pointed to by its index and its subordinates to a file.

386 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX

Write only at run-time, not available at design time.

### Syntax

```
[form.]TList1.Save(index&) = FileHandle%
```

### Remarks

TList1.**Save**(-1) will save the entire contents of the control.

Note the **Save** property depends on the **VisualRoot** property settings

### Example

```
Dim FreeHandle%  
FreeHandle% = FreeFile  
' Open file for output.  
Open "e:\MyTListFile.tlt" For Binary Access Write As FreeHandle%  
TList1.Save(-1) = FreeHandle%  
Close FreeHandle% ' Close file.
```

---

## SaveBuffer Method

### Applies to

TList object

### Description

This function saves a *tree buffer* to a file. Several *tree buffers* may be saved to the same file. The data are stored sequentially.

The return value is zero if the function is successful. Otherwise, the return value is an error code.

Not available at design time.

### Syntax

```
[form.]TList1.SaveBuffer (ByVal hTreeBuffer As Long, ByVal nFile As Integer) As Integer
```

### Example

```
Dim Dummy%, hTreeBuffer%, FreeHandle%  
  
FreeHandle% = FreeFile  
  
' Open file for output.  
Open "c:\MyTListFile.tlt" For Binary Access Write As FreeHandle%  
...  
Dummy% = TList1.SaveBuffer(hTreeBuffer, FreeHandle%)  
  
if Dummy% <> 0 then ..... ' error  
...  
Close FreeHandle% ' Close file
```

### See Also

File/IO in *Using TList* for further information

---

## SaveData Method

### Applies to

TList object

### Description

This method saves TList properties and data to a file compatible with the **LoadData** method. **SaveData** method saves not only TList's tree structure (items), but TList's properties settings like **FontItalic**, **ViewStyle** etc.

---

Files created by **SaveData** can be loaded only by the **LoadData** method, other TList LoadXXX properties won't work here.

---

Files created with **SaveData** method can be viewed and edited by the TDesigner application.

### Syntax

```
[form.]TList.SaveData(FileName As String) As Integer
```

### Remarks

The **SaveData** method returns one of the following values:

Setting	Description
0	Succeeded.
1	File Write Error.
2	File Not Found.

---

## SaveData Method (TListView)

### Description

This method saves TListView properties and data to a file compatible with the TListView's **LoadData** method. **SaveData** method saves not only TListView property settings but also the whole tree structure.

---

Files created by **SaveData** can be loaded only by TListView's **LoadData** method, so TList's LoadData method can't load them.

---

### Syntax

```
[form.]TListView.SaveData(FileName As String) As Integer
```

### Remarks

The **SaveData** method returns one of the following values:

Setting	Description
0	Succeeded.
1	File Write Error.
2	File Not Found.

---

**Note:** The SaveData method doesn't save the images taken from an ImageList, but only references to the images. These references will be used for connecting with corresponding physical images taken from ImageList control when loading TLV files. It is the responsibility of user to save the ImageList data and to have the ImageList available when loading TLV.

---

---

## SaveOne Property

### Applies to

TList object

### Description

This property saves an item pointed to by its index, without its subordinates, to a file.  
Write only at run-time, not available at design time.

## Syntax

```
[form.]TList.SaveOne(index&) = FileHandle%
```

## Remarks

The **SaveOne** property depends on the **VisualRoot** property settings

## Example

```
Dim FreeHandle%
FreeHandle% = FreeFile
' Open file for output.
Open "e:\MyTListFile.tlt" For Binary Access Write As FreeHandle%
TList1.SaveOne(300) = FreeHandle%
Close FreeHandle% ' Close file.
```

---

# SaveSub Property

## Applies to

TList object

## Description

This property saves the subordinates of an item pointed to by its index to a file.  
Write only at run-time, not available at design time.

## Syntax

```
[form.]TList.SaveSub(index&) = FileHandle%
```

## Remarks

The **SaveSub** property depends on the **VisualRoot** property settings

## Example

```
Dim FreeHandle%
FreeHandle% = FreeFile
' Open file for output.
Open "e:\MyTListFile.tlt" For Binary Access Write As FreeHandle%
TList1.SaveSub(300) = FreeHandle%
Close FreeHandle% ' Close file.
```

---

# Scrollbars Property

## Applies to

TList object

## Description

The ScrollBars property determines whether TList scrollbars are displayed. This property is generally set in the design time properties window.  
Read-only at run-time.

## Syntax

```
[form.]TList.Scrollbars
```

## Settings

The **Scrollbars** property settings are:

Setting	Description
0	(Default) Neither scrollbar is displayed.

1	Only the horizontal scrollbar is displayed.
2	Only the vertical scrollbar is displayed.
3	Both scrollbars are displayed.

### Remarks

This property only enables scrollbars, it does not force them to be visible when not needed. By default the specified scrollbars are shown only as required. To show the desired scrollbars all the time set the **DisableNoScroll** property to True.

---

## ScrollHorz Property

### Applies to

TList object

### Description

This property returns or sets the Horizontal scroll position in Percent (%) of the maximum horizontal scroll value).

Run-time only.

### Syntax

[form.]TList.**ScrollHorz**[=int %] **ScrollHPosition** property

### Applies to

TList object

### Description

**ScrollHPosition** property specifies TList's horizontal scroll position in pixels.

### Syntax

TList.**ScrollHPosition** [= pixels]

### Valid Range

0 <= .ScrollHPosition <= .ScrollHRange

### Data Type

Long

### Remarks

To scroll horizontally by some percentage use the ScrollHorz property

### See Also

**ScrollHPosition** property. **ScrollHorz** property

---

## ScrollHRange property

### Applies to

TList object

### Description

**ScrollHRange** property returns the maximum range of horizontal scrolling in pixels – ie: the maximum value to which the ScrollHPosition property may be set.

*Read-only.*

## Syntax

```
[pixels] = TList.ScrollHRange
```

## Data Type

Long

## See Also

**ScrollHPosition** property. **ScrollHorz** property

---

# ScrollVModeMouse, ScrollVModeKeyboard, ScrollVStepMouse and ScrollVStepKeyboard properties

## Applies to

TList object

## Description

These properties control TList's vertical scrolling behavior.

TList may be set to scroll vertically by on a row-by-row basis where TList scrolls up or down by the height of an entire row ( standard behavior ) or to scroll "smoothly" up and down by some specified number of pixels ( specified by ScrollVStepMouse and ScrollVStepKeyboard properties ).

TList may also be set to an "Intelligent" Mixed mode where the choice of row-by-row, or smooth scrolling is made automatically depending on whether the row at the top or bottom of the TList window is too tall to fit fully within the TList window.

The vertical scrolling response to keyboard and mouse actions may be set independently.

The response to keyboard Up/Down arrow keys is specified by the ScrollVModeKeyboard and ScrollVStepKeyboard properties, and the response to mouse clicks on the scrollbar buttons is specified by the ScrollVModeMouse and ScrollVStepMouse properties.

## Applies to

TList object

## Syntax

```
TList.ScrollVModeKeyboard = enum  
TList.ScrollVModeMouse = enum  
TList.ScrollVStepMouse = pixels  
TList.ScrollVStepKeyboard = pixels
```

## Settings

The two properties **ScrollVModeKeyboard** and **ScrollVModeMouse** may each be independently set to any of the following three values:

Constant	Value	Description
TL_SCROLLVMODE_STANDAR D	0	<b>Standard</b> mode (Default). TList will scroll vertically in the same manner as most List/Tree/Grid controls – on a row by row basis. If a row is too tall for complete display within the control window, TList will scroll by jumping to the next item and the bottom of very tall items may never be seen (except as displayed in a tooltips in response to mouse hovering over the item).
TL_SCROLLVMODE_SMOOTH	1	<b>Smooth</b> mode. TList will scroll vertically by a specified number of pixels, using settings of the

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

TL\_SCROLLVMODE\_MIXED                      2

TList.ScrollVStepMouse (for response to Mouse clicks) and TList.ScrollVStepKeyboard (for response to keyboard Up/Down arrows) properties.

**Intelligent/Mixed** mode.

Only the rows that are too tall to be fully displayed within the client view area of TList will be vertically scrolled smoothly, all other rows will be scrolled in the standard row-by-row mode.

Thus TList will scroll normally row by row until coming to an item which is too tall for display within the control window, at which point it will switch to Smooth Scrolling mode, and then upon reaching the next item that fits within the control window, TList will switch back to standard row-by-row scrolling.

The two properties **ScrollVStepMouse** and **ScrollVStepKeyboard** may each be independently set to an integer number representing the number of pixels by which the control will scroll vertically in **Smooth** mode or for tall rows in **Mixed** mode.

**Defaults**

TList.ScrollVModeKeyboard = TL\_SCROLLVMODE\_STANDARD

TList.ScrollVModeMouse = TL\_SCROLLVMODE\_STANDARD

TList.ScrollVStepMouse = 10 pixels

TList.ScrollVStepKeyboard = 10 pixels

**Example**

```
TList.ScrollVModeMouse = TL_SCROLLVMODE_SMOOTH
TList.ScrollVStepMouse = 10
```

---

## ScrollVPosition property

**Applies to**

TList object

**Description**

The **ScrollVPosition** property specifies the Vertical Scroll position of TList measured in visible items.

Only Visible Items ( items not collapsed, and not identified as Hidden or of 0 height) are counted.

**Syntax**

```
TList.ScrollVPosition [= Nth Visible Item]
```

**Valid Range**

0 <= .ScrollVPosition <= .ScrollVRange

**Data Type**

Long

**Remarks**

To scroll to a specific item using it's ItemIndex (counting all items rather than counting just visible items) use the TopIndex property.

**See Also**

**ScrollVRange** property, **TopIndex** property



---

## ScrollVRange property

### Applies to

TList object

### Description

**ScrollVPosition** property returns the maximum range of Vertical scrolling– ie: the maximum value to which the ScrollVPosition property may be set.

---

Note: scrolling range is measured in items.  
Read-only.

---

### Syntax

```
[items] = TList ScrollVRange
```

### Data Type

Long

### See Also

**ScrollVPosition** property, **TopIndex** property

---

## SelBackColor and SelForeColor Properties

### Applies To

TList control

TListCellDef object

### Description

When applied to a TList control:

- **SelBackColor** determines the background color of a selected item.
- **SelForeColor** determines the foreground color used to display text in a selected item.

When applied to a TListCellDef object:

- **SelBackColor** determines the background color of a selected cell.
- **SelForeColor** determines the foreground color used to display text in a selected cell.

### Syntax

```
[form.]TList.SelBackColor[ = color&]  
[form.]TList.SelForeColor[ = color&]  
[form.]TList.Grid.Cells(Row&, Col&).SelBackColor[ = color&]  
[form.]TList.Grid.Cells(Row&, Col&).SelForeColor[ = color&]
```

### Remarks

The default settings at design time are:

```
SelBackColor = HIGHLIGHT 'system default backcolor for selected items  
SelForeColor = HIGHLIGHT_TEXT ' system forecolor for selected items
```

For the value of these constants, refer to the CONSTANT.TXT file supplied with *Visual Basic*.  
Setting of these properties updates the control unless the **Redraw** property is set to False.

### Data Type

Long

---

## SelBorderStyle property (TListCellDef object)

### Applies To

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •  
393

**TListCellDef** object

### Description

**TListCellDef**'s **SelBorderStyle** property determines how borders will be drawn around a cell when it becomes selected.

### Syntax

```
[form.]TListCellDef.SelBorderStyle [= enum%]
```

### Settings

**SelBorderStyle** property's settings are:

Constant	Setting	Description
TLBORD_DEFAULT_INV	-1	(Default) Perform the default inversion procedure (when the cell is being selected) over the border specified by <b>BorderStyle</b> property.
TLBORD_NONE	0	No border.
TLBORD_SINGLE	1	Single one-pixel border.
TLBORD_DOUBLE	2	2-pixel border.
TLBORD_INSET	3	3-D inset border.
TLBORD_OUTSET	4	3-D outset border.

### Example

```
'set the selected single border style for a particular cell in the grid
TList1.Grid.Cells(5, 7).CellDef.SelBorderStyle = TLBORD_SINGLE

'set the selected double border style for a particular cell in the ItemGrid belonging to item 30
TList1.ItemGrid(30).Cells(5, 7).CellDef.SelBorderStyle = TLBORD_DOUBLE

'set the selected double border style for a particular cell in the ItemGrid belonging to item 30
TList1.ItemCell(30).SelBorderStyle = TLBORD_INSET
```

### Data Type

Integer

---

## SelBorderColor property (TListCellDef object)

### Applies To

**TListCellDef** object

### Description

Determines the default border color displayed around a cell when it becomes selected.

### Syntax

```
TListCellDef.SelBorderColor[ = color&] - sets selected border color for a specific cell.
```

### Remarks

This property value will be ignored when **SelBorderStyle** = **TLBORD\_DEFAULT\_INV**.

### Example

```
'set the selected border color for a particular cell in the grid
TList1.      Grid.Cells(5, 7).CellDef.SelBorderColor = RGB(255, 0, 0)
```

### Data Type

## Selectable Property (TListCellDef object)

### Applies to

TListCellDef object

### Description

The Selectable property provides control over the end-user's ability to select specific List/Tree items, or Cells, Rows, Columns in a Grid.

This property is helpful for specifying non-selectable regions in the grid or tree. By default this property set to **True** for all TListCellDef objects.

### Syntax

```
[TListCellDef].Selectable = [boolean]
```

### Example

```
'makes third column in the main grid non-selectable
TList1.Grid.ColDefs(3).CellDef.Selectable = False
' ...
'makes the cell with coordinates row=2 column=4 in Item 30's ItemGrid non-selectable
TList1.ItemGrid(30).Cells(2,4).CellDef.Selectable = False
' ...
'makes the item with index 100 non-selectable
TList1.ItemCell(100).Selectable = False
```

### Remarks

Use the **Activatable** property of the TListCellDef object to control the end-user's ability to navigate to any particular item/cell/row using keyboard, mouse or using properties. Non-Selectable elements may not be selected by either end-user or program control.

## Selected Property

### Applies to

TList object

TListNodes object

### Description

This property determines the selection status of an item in a list box. This property is an array of Boolean values with the same number of items as the **List** property. You can set this property only while in multiple selection mode, defined by the **MultiSelect** property. With **MultiSelect** set False, setting of **Selected** has no effect and the return value is always False.

Not available at design time.

### Syntax

```
[form.]TList.Selected(index&)[ = {True|False}]
[form.]TList.Nodes(index&).Selected[ = {True|False}]
```

### Settings

The **Selected** property settings are:

Setting	Description
True	The item is selected.

False	(Default) The item is not selected.
-------	-------------------------------------

### Remarks

This property is particularly useful when users want to make multiple selections. You can quickly check if an item in the list is selected. You can also use this property to select or deselect items in a list.

If only one item is selected, you can use the **ListIndex** property to get the *index* of the selected item. However, in a multiple selection, the **ListIndex** property returns the *index* of the item contained within the focus rectangle, whether or not the item is actually selected.

Only visible items can be selected. An item whose parent is collapsed can not be selected.

Setting of **Selected** property updates the control unless the **Redraw** property is set to False.

### Data Type

Boolean

### See Also

**SellItemCount** and **SellItemIndex** properties which return the number of selected items and the indexes of all selected items

---

## Selected Property (TListGridCell, TListRowDef and TListColDef objects)

### Applies to

TListGridCell, TListRowDef, TListColDef objects

### Description

This property returns the current status of the corresponding element (cell, row or column). Read-only. To select cell, row or column add an object to the correspondent collection of TListGrid object (**SelectedCells**, **SelectedRows**, **SelectedColumns**).

### Syntax

```
[boolean] = [TListGridCell].Selected
[boolean] = [TListColDef].Selected
[boolean] = [TListRowDef].Selected
```

### Example

```
Dim bStatus As Boolean
'Select the 3rd column in the main grid
TList1.Grid.SelectedColumns.Add TList1.Grid.ColDefs(3)
'...
'read back the status of the column
bStatus = TList1.Grid.ColDefs(3).Selected
'...
'Select the cell in Row 2, Column 4
TList1.Grid.SelectedCells.Add TList1.Grid.Cells(2,4)
'...
'read back the status of the cell
bStatus = TList1.Grid.Cells(2,4).Selected
'...
'select row 10 in the main grid
TList1.Grid.SelectedRows.Add TList1.Grid.Rows(10)
'...
'read back the status of the row
bStatus = TList1.Grid.RowDef(10).Selected
```

### Remarks

- A Row is considered selected if all selectable cells in the row are selected (at least one cell in the row should be selectable).
- A Column is considered selected only if all selectable cells in the column are selected (at least one cell in the column should be selectable).

### See Also

Selectable property

---

## SelectedCells Property (TListGrid object)

### Applies to

TListGrid object

### Description

This property returns a reference to a collection of TListGridCell objects that are currently selected. See TListSelectedGridCells object

### Syntax

```
[TListSelGridCells] = [TListGrid].SelectedCells
```

---

## SelectedColumns Property (TListGrid object)

### Applies to

TListGrid object

### Description

This property returns a reference to a collection of TListColDef objects that are currently selected. See TListSelectedGridColumnns object

### Syntax

```
[TListSelGridColumnns] = [TListGrid].SelectedColumns
```

---

## SelectedRows Property (TListGrid object)

### Applies to

TListGrid object

### Description

This property returns a reference to a collection of TListRowDef objects that are currently selected. See TListSelectedGridRows object

### Syntax

```
[TListSelGridRows] = [TListGrid].SelectedRows
```

---

## SelectedItem Property (TListTreeView)

### Description

With the SelectionMode property set to 0 (single select mode), this property works just like the MS TreeView property of the same name. It returns the item with the focus, which in single select mode is the selected node

In multi-selection mode, the **SelectedItem** property returns the node with the focus regardless of whether that node is actually selected, and regardless of whatever other nodes are also selected.

**TLTreeView.SelectedItem** property always points to the same item as **TList's ListIndex** property (the item with the focus), even if this item is NOT selected.

---

## SelectedNodes Property (TListTreeView)

### Description

This property returns a **TVwSelectedNodes** collection containing all selected nodes in **TListTreeView**.

### Syntax

```
TVwSelectedNodes = [form.]TListTreeView.SelectedNodes
```

### Example

```
'Select two nodes (first and third ones) using Add method of TVwSelectedNodes collection
Dim objSelectedNodes as TVwSelectedNodes
Set objSelectedNodes = TListTreeView.SelectedNodes
objSelectedNodes.Add TListTreeView.Nodes(1)
objSelectedNodes.Add TListTreeView.Nodes(3)

'Deselect one node using Remove method of TVwSelectedNodes collection
TListTreeView.SelectedNodes.Remove TListTreeView.Nodes(SomeIndex)
Or
TListTreeView.SelectedNodes.Remove SelectedNodeIndex

'Deselect the last Selected Node
With TListTreeView.SelectedNodes
.Remove .item( .Count-1 )
End With

'Enumerate all selected items and display them in debug window
Dim objNode as Node
For Each objNode In TListTreeView.SelectedNodes
Debug.Print " Node Text= " & objNode.Text
Next
```

### Remarks

Selected nodes are indexed starting with 1 as with other **TLTreeView** node indexing ( compatible with **MSTreeview** indexing standards). This is different than the 0 based indexing used by **TLTreeView.TList** nodes and items.

---

## SelectEx Property

### Applies to

**TList** object

### Description

Setting the **SelectEx** property selects all subordinate elements of an item specified by its index. If the index is set to -1, it selects the entire list (as defined by the **CurrentIndexMethod** property). Not available at design time, write-only at run time.

### Syntax

```
[form.]TList.SelectEx(index&) [ = {True|False}]
```

## Example

```
'Copy all subordinates of item 5 to a second  
' instance of TList  
TList1.CurrentIndexMethod = 2  
TList1.SelectEx(5) = -1  
Tree_buffer& = TList1.CopySelected  
TList2.Add (-1) = Tree_buffer&  
TList2.FreeBuffer(Tree_buffer&)
```

## Data Type

Boolean

---

# SelectionMode property (TListGrid object)

## Applies to

TList object

## Description

The **SelectionMode** property controls the how objects (rows and/or cells) are selected within a particular TListGrid object. This can apply to either an Item Grid or for the primary TList Grid containing the complete data set.

There are three selection modes:

1. **Inherited selection** - In this mode selection within a grid is handled as if the rows are just ordinary items within the main list / tree. The selection mode is then specified by the **MultiSelect** property of the TList object (see **MultiSelect** property for details).
2. **Single Selection** - In this selection mode the user can select only **one** row or **one** cell at a time from the specified Grid. (Selection of Rows or Cells is dependant on the **ActivationMode** property of the grid)

The setting of the TList MultiSelect property is overridden within the Grid (even if TList.MultiSelect allows multiple selection, only a single element of the grid may be selected).

When applied to an ItemGrid, the selection in this mode is completely independent from the rest of the TList structure (items outside the ItemGrid or within other ItemGrids).

When applied to an ItemGrid, the selection status will be preserved when the user leaves a grid and moves to an object outside the grid using keyboard or mouse. Note that if TList.MultiSelect is False ( allowing only a single selection outside the grid, but an ItemGrid's SelectionMode is set to SingleSelection, this may result in multiple TList items being selected – a maximum of one outside any item grids, and a maximum of one inside each ItemGrid.

In order to select a row or a cell, the end user may press the <SPACE> key on the keyboard while the focus is on the cell (while the desired row/cell is active – as marked with focus rectangle), or single click on the desired cell or row using the mouse. Any previous selection in the same item grid is automatically deselected when a new element is selected. (Deselection of elements outside the grid may be depends on the setting of TList's Multi-Select Mode)

The activation mode (the ability to set the focus upon individual rows, individual cells or a combination of rows and cells) is controlled via the **ActivationMode** property of the Grid.

3. **Multiple Selection** - In this selection mode the user can select several rows or cells from the specified grid. (Selection of Rows or Cells is dependant on the **ActivationMode** property of the grid)

As with Single Selection mode, the selection is completely independent from the rest of the tree structure or other grid object. Even if TList.MultiSelect is False and only allows Single Selection outside the Grid, multiple items may be selected inside a Grid.

The selection status will be preserved when user leaves a grid object and moves to another part of the tree using keyboard or mouse.

When applied to an ItemGrid, the selection status will be preserved when user leaves a grid and moves to an object outside the grid using keyboard or mouse.

- **Keyboard interaction:**

To select/deselect a row or a cell, the user has to press SPACE key while the desired element is activate (marked with focus rectangle). The behavior is similar to that set for ordinary List / Tree items ( outside a grid) using the Simple Multiple Selection mode of the TList.**MultiSelect** property.

To select a set of the rows or cells, the user can hold SHIFT key and use Up/Down/Left/Right arrow keys. Note: in case of using SHIFT+arrow key combination any previous selection (in the same grid) will be discarded and the new selection extents from the previous active row/cell to the currently active one.

- **Mouse interaction:**

When the user clicks using the mouse over a cell/row, the previous selection status will be discarded and only the element being clicked will be selected.

When the user holds SHIFT key and clicks over a cell / row, the previous selection (in the same grid) will be discarded and the new selection will extend from the previously active row/cell to the current one (The same as the SHIFT + Arrow Key keyboard behavior).

When the user holds CTRL key and clicks over a cell / row, only the element being clicked changes the selection status, the status of other elements remains unchanged ( the same behavior as for when a user hits the arrow key without holding SHIFT and then selects/deselects row/cell by pressing SPACE key).

### **Selection Modes comparison table**

This is a list of differences in the behavior of the control between Selection modes.

<b>User Action</b>	<b>Selection Mode = Inherited</b>	<b>Selection Mode = Single or Multiple Selection</b>
<b>Clicking item grid column titles (for tree item that has an</b>	The item and the Column Titles of the child ItemGrid are both selected.	Item grid column titles are NOT selected



<b>ItemGrid as a child)</b>		
<b>Selecting a row in ItemGrid object</b>	The entire row is selected, including the row title cell.	The row title cell is NOT selected unless TL_SELOPT_MARKSELECTED_ROWCOLTITLE flag is set in the <b>SelectionOptions</b> property.

### Syntax

`TList.SelectionMode [= enum%]`

### Remarks

**SelectionMode** property's settings are:

Constant	Setting	Description
TL_SELMODE_INHERITED	-1	(Default) The grid object inherits the selection mode from the tree itself (controlled by <b>MultiSelect</b> property).
TL_SELMODE_SINGLE	0	A click or the spacebar key press selects a single row/cell in the grid.
TL_SELMODE_MULTIPLE	1	A click or the spacebar key press selects or deselects a row/cell in the grid. SHIFT+click or Shift+arrow key extends the selection from the previously activated row/cell to the current one (previous selection gets removed automatically). CTRL+click selects or deselects a row/cell in the grid.

### Data Type

Integer

### See Also

**SelectionOptions** property

---

## SelectionOptions Property (TListGrid object)

### Applies to

**TListGrid** object

### Description

This property specifies some options that allow user to control the selection process more thoroughly.

### Syntax

`[TListGrid].SelectionOptions = [enum]`

### Settings

The **SelectionOptions** property is a bit-field valued property set by OR'ing the desired combination of flag values:

Constant	Setting	Description
TL_SELOPT_SELECTROWS	&H1	A whole grid row will be selected when user

_ONTITLECLICK		clicks over the corresponding row title. (Not enabled in Cell by Cell activation mode and selection mode = TL_SELMODE_SINGLE)
TL_SELOPT_SELECTCOLS _ONTITLECLICK	&H2	A whole grid column will be selected when user clicks over the corresponding column title. (NOT supported if ActivationMode = Item or Row. NOT supported unless in multiple selection mode - for Tree and/Or Grid)
TL_SELOPT_SELECTGRID _ONCORNERCLICK	&H4	A whole grid (all cells) will be selected when user clicks over the corner cell of the grid. (NOT supported unless in multiple selection mode - for Tree and/Or Grid.)

## SelectionMode Property (TListTreeView)

### Description

This property determines the selection mode of the TListTreeView component and supports single selection or one of two multi-selection modes (the same selection modes as are supported using the MultiSelect property of TList).

### Syntax

[form.]TListTreeView.SelectionMode [= {SelectionMode}]

### Settings

The **SelectionMode** property settings are:

Setting	Value	Description
TLMULSEL_NONE	0	(Default)Multiple selections are not allowed.
TLMULSEL_SIMPLE	1	Simple multiple selection. A click or the Spacebar selects or deselects an item in the list.
TLMULSEL_EXT	2	Extended multiple selection. Shift+click or Shift+arrow key extends the selection from the previously selected item to the current directory. Ctrl+click selects or deselects an item in the list.
TLMULSEL_EXPLORER_LIKE	3	Windows Explorer style multiple selection support. The selection is changed on MouseDown for all situations except the case when user holds CTRL key while clicking the mouse or when clicking over an already selected item.

### Data Type

Integer

### Remarks

Setting the SelectionMode property of the TListTreeView object automatically updates the MultiSelect property of the TList object within the TListTreeView (TListTreeView.TList.MultiSelect). Likewise setting the MultiSelect property of the TList automatically updates the TListTreeView SelectionMode property

Setting the SelectionMode property to 0 (Single Selection) automatically deselects all nodes except the node with the focus (the node pointed to by the SelectedItem property).

In multiple selection modes, The TListView SelectedItem property refers to the item in the tree that has a focus - this is the same as TreeView.SelectedNodes(1).

---

## SellItemCount Property

### Applies to

TList object

### Description

The **SellItemCount** property returns the number of currently selected items.  
Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.SellItemCount
```

### Data Type

Long

---

## SellItemIndex Property

### Applies to

TList object

### Description

The **SellItemIndex** property contains an array of the indices of all selected items. Reading this property returns the index of the N<sup>th</sup> selected item. Not available at design time, read-only at run time.

### Syntax

```
[form.]TList.SellItemIndex(IndexOfTheSelectedItem&)
```

### Example

```
Text1.Text = ""  
For I = 0 To TList1.SellItemCount - 1  
    ItemIndex& = TList1.SellItemIndex(I)  
    Text1.SelText = TList1.List(ItemIndex&) & Chr$(13) & Chr$(10)  
    Text1.SelLength = 0  
Next
```

### Data Type

Long

---

## SelStart And SelLength Properties (TListEditingChangeInfo object)

### Applies to

TListEditingChangeInfo object

### Description

**SelStart** property returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.

**SelLength** property returns or sets the number of characters selected in the edit area of **TListTextBox**, **TListComboBox** with editing area, **TListSpin** objects.

These properties are available only through **TListEditingChangeInfo** object inside **GridCellEditingChange** and **ItemEditingChange** events.

Note: Setting **SelLength** less than 0 will be interpreted as the value = 0 was set instead. Setting **SelStart** greater than the text length sets the property to the existing text length; changing **SelStart** changes the selection to an insertion point.

### Syntax

```
[TListEditingChangeInfo].SelStart = [Long]
[TListEditingChangeInfo].SelLength = [Long]
```

### Example

```
Private Sub TList1_ItemEditingChange(ByVal ItemIndex As Long, _
    ByVal objChangeInfo As TListEditingChangeInfo)

If objChangeInfo.ValueType = TLED_CHANGE_COMBOBOX_EDITAREA
    Dim strUserEnteredData as String
    StrUserEnteredData = objChangeInfo.Value ' string typed by user
    Dim iDataLen as Long
    iDataLen = Len( StrUserEnteredData ) ' length of string typed by user
    Dim objItem as TListComboItem
    For Each objItem In objChangeInfo.EditInfoObject.ComboBox.Items
        If Left$(objItem.CellValue, iDataLen) = StrUserEnteredData _
            Then ' match found – set full text of list item in text area
            objChangeInfo.Value = objItem.CellValue
            ' select the portion of the text we have just added
            objChangeInfo.SelStart = iDataLen
            objChangeInfo.SelLength = Len(objChangeInfo.Value)-iDataLen
        Exit For
    End If
Next
End If
End Sub
```

### See Also

**GridCellEditingChange** and **ItemEditingChange** events, **ValueType**, **Value**, **OldValue**, **OldSelStart**, **OldSelLength**, **EditInfoObject** properties

---

## SetFocus Method

### Applies to

**TList** object

### Description

This method sets the focus to TList control.

### Syntax

```
[form.]TList.SetFocus
```

### Remarks

For more information, see the description of the **SetFocus** method in the *Microsoft Visual Basic Language Reference*.

---

## Shift Property

### Applies to

TList object

### Description

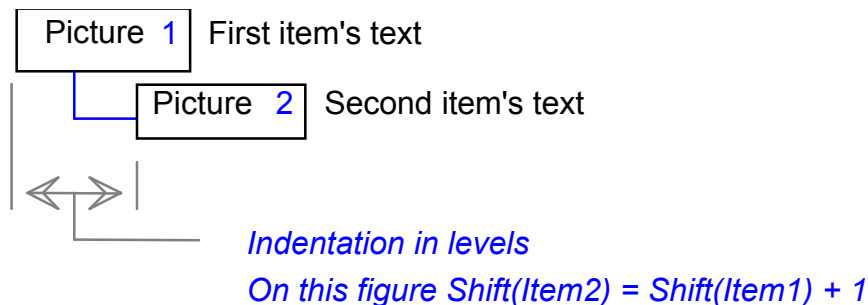
The **Shift** property is an integer array in which each element defines an item's hierarchic indentation. This property is not available at design time.

### Syntax

```
[form.]TList.Shift(index&)[ = new_shift%]
```

### Remarks

Refer to the illustration below.



The **Shift** property may only be increased if the item immediately above is a peer (has the same current shift value) or has a higher **Shift** value. Incrementing the **Shift** property by one moves the item to become an immediate subordinate of its previous peer, simultaneously increasing the shift value of all the item's children.

Decrementing the **Shift** property by one moves an item to the same level as that of its parent, (keeping all the subordinate items of the shifted item as children and modifying their **Shift** properties appropriately). The shifted item now shares the same parent as its previous parent.

The **Shift** property is TList's analog to MSOutline's **Indent** property.

The **Shift** property doesn't depend on the **VisualRoot** property settings

### Example

```
Sub Form_Load ()
    TList.AddItem "Fred"
    TList.AddItem "Fred's son"
    TList.AddItem "Fred's Grand daughter", 0
    TList.Shift(2) = 1
    TList.Shift(1) = 1
End Sub
```

### Data Type

Integer (Array)

---

## ShiftStep Property

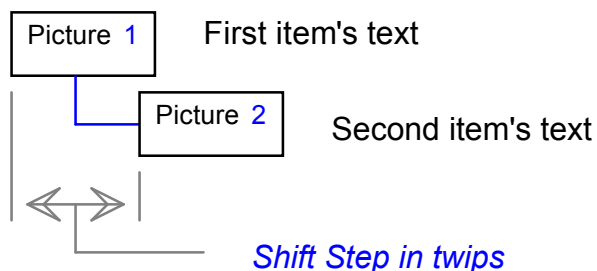
### Applies to

TList object

### Description

This property determines the horizontal indentation between items one indentation level apart. Measured in twips.

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •



### Syntax

```
[form.]TList.ShiftStep [= long_expression&]
```

### Remarks

Setting of the **ShiftStep** property updates the control unless the **Redraw** property is set to False.

### Data Type

Long

## ShowCaption Property

### Applies to

TList object

### Description

Setting the **ShowCaption** property determines the style of the control's caption.

### Syntax

```
[form.]TList.ShowCaption [= enum%]
```

### Settings

The **ShowCaption** property settings are:

Setting	Description
0	(Default) None.
1	Simple.
2	3D.

### Data Type

Integer

## ShowChildren Property

### Applies to

TList object

### Description

This property determines whether TList will scroll upon expansion of an item, to show as many newly visible subordinate items as possible.

### Syntax

```
[form.]TList.ShowChildren[=bool_expr%]
```

### Settings

The **ShowChildren** property settings are:

Setting	Description
True	(Default) TList will scroll automatically during the <b>Expand</b> event.
False	TList will not automatically scroll during the <b>Expand</b> event.

### Data Type

Boolean

## ShowHiddenItems Property

### Applies to

TList object

### Description

This property makes all "always hidden items" visible without changing any of **ItemAlwaysHidden** properties or **MarkedItemsAlwaysHidden** property.

Not available at design time.

### Syntax

```
[form.]TList.ShowHiddenItems [= bool%]
```

### Data Type

Boolean

## ShowColumnTitles Property

### Applies To

TListReport object

### Description

This property specifies if Tree Grid headers should be printed on every page.

Not available at design-time.

### Syntax

```
TListReportObject.ShowColumnTitles [= bool%]  
[form.]TList.Report.ShowColumnTitles [= bool%]
```

### Data Type

Integer (Boolean)

## ShowColTitles Property

### Applies To

TListGrid object

### Description

This property determines whether column titles are displayed in a grid. If set to True column titles will be shown.

### Syntax

```
TListGridObject.ShowColTitles [= bool%]  
[form.]TList.Grid.ShowColTitles [= bool%]  
[form.]TList.ItemGrid(ItemIndex&). ShowColTitles [= bool%]
```

### Data Type

Boolean

---

## ShowRowTitles Property

### Applies To

TListGrid object

### Description

This property determines whether row titles are displayed in a grid. If set to True row titles will be shown.

### Syntax

```
TListGridObject.ShowRowTitles [= bool%]  
[form.]TList.Grid.ShowRowTitles [= bool%]  
[form.]TList.ItemGrid(ItemIndex&).ShowRowTitles [= bool%]
```

### Data Type

Boolean

---

## ShowTitles Property

### Applies to

TList object

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## SmartDragDrop Property

### Applies to

TList object

### Description

This property determines whether TList will change the selection of the currently drag-dropped item in response to a mouse click in multi-selection mode.

### Syntax

```
[form.]TList.SmartDragDrop[=bool_expr%]
```

### Settings

The **SmartDragDrop** property settings are:

Setting	Description
True	TList will change the selection on <b>MouseUp</b> event.
False	(Default) TList will change the selection on <b>MouseDown</b> event.

### Remarks

When this property is set to True the changing of selection of the item occurs on **MouseUp** event, not on **MouseDown** as would otherwise be the case. The **SmartDragDrop** property has no effect on TList behavior when **MultiSelect** property is set to 0 (None).

The following describes how events are generated when **SmartDragDrop** property is set to True:

- User clicks on the item - **ListIndex** is changed and focus rectangle appears;
- MouseDown** event is generated - no **Click** events are triggered;



- c) **MouseMove** events are generated as appropriate;
- d) User releases the mouse button;
- e) if **Drag** method was called inside the **MouseDown** event or **MouseMove** event neither **MouseUp** nor **Click** is generated;
- f) if **Drag** method wasn't called inside the **MouseDown** event or **MouseMove** event; first the **Click**(**PictureClick**, **MarkClick**) event and then **MouseUp** event is generated.

X and Y parameters of **MouseDown** events are expressed in PIXELS when **SmartDragDrop** property is set to True, not in terms of container's scale as would otherwise be the case. See the **MouseDown** event description.

### Data Type

Boolean

---

## ShowTooltip Event

### Applies to

TList object

### Description

This event is triggered before display of TList's tooltip window.

By setting the properties of the parameter ( an object ) the programmer may exercise control over the content and presentation of the tooltip.

### Syntax

TList.ShowTooltip(objTooltipArgs As TListShowTooltipArgs)

### Parameters

Parameter	Description
ObjToolTip	<p>An object of type TListShowTooltipArgs that defines the content and presentation of a tooltip.</p> <p>The TListShowTooltipArgs object has the following properties:</p> <ul style="list-style-type: none"> <li>• <b>Cancel</b> - set this property to True to prevent the tooltip window from being displayed</li> <li>• <b>Tooltip</b> - returns a <b>TListTooltip</b> object specifying the content of the tooltip window to be displayed</li> <li>• <b>ItemIndex</b> - returns the item index of the item under the mouse</li> <li>• <b>GridCell</b> - if Mouse is over a grid cell, this property returns an instance of a <b>TListGridCell</b> object. Otherwise this property returns nothing.</li> <li>• <b>MouseScreen</b> - returns a <b>TListPoint</b> object with read only properties that identify the current mouse position relative to the top left corner of the overall windows display, in pixels.</li> <li>• <b>MouseLocal</b> - returns a <b>TListPoint</b> object with read only properties that identify the current mouse position relative to the top left corner of the TList control, measured in pixels</li> <li>• <b>Window</b> - returns a <b>TListTooltipWindow</b> object</li> </ul>

### Example

Specifies the tooltip content and color at the time tooltip is to be shown:

```
Sub TList1_ShowTooltip ( ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
  With objTooltipArgs
    If .GridCell Is Nothing Then
      'for standard items show ItemTag as Tooltip
      .ToolTip.Text = TList1.ItemTag(.ItemIndex)
      .ToolTip.Style.BackColor = RGB(255, 100, 100)
    End If
  End With
End Sub
```

```

Else
    'for Grid Cells show the cell's Tag as a Tooltip
    .ToolTip.Text = .GridCell.CellDef.Tag
    .ToolTip.Style.BackColor = RGB(100, 100, 255)
End If
End With
End Sub

```

Cancelling the tooltip for grid cells if a checkbox is unchecked.

Note: tooltips will be shown for the data outside of a grid such as parent items of ItemGrid objects.

```

Sub TList1_ShowTooltip ( ByVal objTooltipArgs As TListProLibCtl.TListShowTooltipArgs)
    With objTooltipArgs
        If chkShowGridCellToolTips.Value = 0 and Not (.GridCell Is Nothing ) Then
            .Cancel = True
        End If
    End With
End Sub

```

## Sorting Property (TListComboItems Object)

### Applies to

**TListEditInfo** object

### Description

This property sets the sorting order for the items in the **TListComboBox** drop-down list and in the collection

### Values

The **Sorting** property settings are:

Constant	Value	Description
TLCOMBO_SORT_NONE	0	<b>(Default Value)</b> The items are not sorted. Their order is initially defined by the order they are added to the list
TLCOMBO_SORT_ASCENDING	1	Items are sorted in ascending order.
TLCOMBO_SORT_DESCENDING	2	Items are sorted in descending order.

### Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.Sorting [= enum%]
```

### Example

```

[TListEditInfo].ComboBox.Items.Add 1,, "One"
[TListEditInfo].ComboBox.Items.Add 2,, "Two"
[TListEditInfo].ComboBox.Items.Add 3,, "Three"
[TListEditInfo].ComboBox.Items.Add 4,, "Four"
'sort all items in ascending order by display value
[TListEditInfo].ComboBox.Items.Sorting = TLSORT_ASCENDING
[TListEditInfo].ComboBox.Items.SortingKey = COMBO_SORT_BY_DISPLAY_VALUE

```

---

## SortingKey Property (TListComboltems Object)

### Applies to

TListEditInfo object

### Description

This property sets the value according to which the items in the ComboBox drop-down list are sorted.

### Values

The **SortingKey** property settings are:

Constant	Value	Description
COMBO_SORT_BY_VALUE	1	Sort by CellValue.
COMBO_SORT_BY_DISPLAY_VALUE	2	(Default) Sort by DisplayValue. If DisplayValue is not specified CellValue will be used instead.

### Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.SortingKey [= enum%]
```

### Example

```
[TListEditInfo].ComboBox.Items.Add 1,, "One"  
[TListEditInfo].ComboBox.Items.Add 2,, "Two"  
[TListEditInfo].ComboBox.Items.Add 3,, "Three"  
[TListEditInfo].ComboBox.Items.Add 4,, "Four"  
[TListEditInfo].ComboBox.Items.Sorting = TLSORT_ASCENDING  
[TListEditInfo].ComboBox.Items.SortingKey = COMBO_SORT_BY_DISPLAY_VALUE
```

---

## SortingStyle Property (TListComboltems Object)

### Applies to

TListEditInfo object

### Description

This property provides additional control over the sorting of the ComboBox drop-down list.

### Values

The **SortingStyle** property may be set to a combination (OR) of the following values:

Constant	Value	Description
TL_B_SORT_IGNORE_CASE	&H1	If set, ignores the case while sorting.
TL_B_SORT_NUMBERS_SORT	&H2	If set, treats string representation of numbers as numbers, ie: "10" comes after "2".
TL_B_SORT_STRING_SORT	&H4	If set, treats punctuation the same way as symbols.

## Syntax

```
[TListEditInfo].EditInfo.ComboBox.Items.SortingStyle [= enum%]
```

## Example

```
[TListEditInfo].ComboBox.Items.Add 1,, "1"  
[TListEditInfo].ComboBox.Items.Add 2,, "2"  
[TListEditInfo].ComboBox.Items.Add 10,, "10"  
[TListEditInfo].ComboBox.Items.Add 20,, "20"  
[TListEditInfo].ComboBox.Items.Sorting = TLSORT_ASCENDING  
[TListEditInfo].ComboBox.Items.SortingKey = COMBO_SORT_BY_DISPLAY_VALUE  
[TListEditInfo].ComboBox.Items.SortingStyle = TL_B_SORT_NUMBERS_SORT
```

---

# Sorted, SortingMethod, SortingStyle, and SortingKey Properties

## Applies To

TListGrid object

TListLevelDef object

TListNode object

## Description

These properties provide control over the sorting applied to a Grid, a Heirarchic Level, or children of a Node object.

- The **ItemSorted** property initiates, halts or resets the sorting.
- The **ItemSortingMethod** property specifies ascending or descending sort order.
- The **ItemSortingStyle** property provides additional control such as Numeric or Case Sensitive Sorting.
- The **ItemSortingKey** property specifies what data (visible text or hidden ItemValues) should be used as the key for the sorting.

## Syntax

```
TListGridObject.Sorted [= Enum%]  
TListGridObject.SortingMethod [( SortPriority) ] [= enum%]  
TListGridObject.SortingKey [( SortPriority) ] [= ValueName$]  
TListGridObject.SortingStyle [( SortPriority) ] [= settings&]  
  
[form.]TList.Grid.Sorted [= bool%]  
[form.]TList.Grid.SortingMethod [( SortPriority) ] [= enum %]  
[form.]TList.Grid.SortingKey [( SortPriority) ] [= ValueName$]  
[form.]TList.Grid.SortingStyle [( SortPriority) ] [= settings&]  
  
[form.]TList.ItemGrid(ItemIndex&).Sorted [= bool%]  
[form.]TList.ItemGrid(ItemIndex&).SortingMethod [( SortPriority) ] [= enum %]  
[form.]TList.ItemGrid(ItemIndex&).SortingKey [( SortPriority) ] [= ValueName$]  
[form.]TList.ItemGrid(ItemIndex&).SortingStyle [( SortPriority) ] [= settings&]  
  
TListLevelDefs( level&).Sorted [= bool%]  
TListLevelDefs( level&).SortingMethod [( SortPriority) ] [= enum %]  
TListLevelDefs( level&).SortingKey [( SortPriority) ] [= ValueName$]  
TListLevelDefs( level&).SortingStyle [( SortPriority) ] [= settings&]  
  
TListNodeObject.Sorted [= Enum%]  
TListNodeObject.SortingMethod [( SortPriority) ] [= enum%]  
TListNodeObject.SortingKey [( SortPriority) ] [= ValueName$]  
TListNodeObject.SortingStyle [( SortPriority) ] [= settings&]  
  
[form].Node(ByVal ItemIndex as Long).Sorted [= Enum%]  
[form].Node(ByVal ItemIndex as Long).SortingMethod [( SortPriority) ] [= enum%]  
[form].Node(ByVal ItemIndex as Long).SortingKey [( SortPriority) ] [= ValueName$]
```

```
[form].Node(ByVal ItemIndex as Long).SortingStyle [( SortPriority) ] [= settings&]
```

### Syntax Change:

- **ItemSorted** constants have been changed.
- An empty or unassigned **ItemSortingKey** value previously indicated sorting by the old (obsolete) **ItemXXXValue** properties. In TList 8, an empty or unassigned **ItemSortingKey** indicates sorting by the visible text. To sort by an **ItemXXXValue** property set the **ItemSortingKey** to "\_\_ObsoleteValue".
- **ItemSortingMethod** introduced to specify Ascending or Descending sorting order.
- Data may be sorted by a prioritized list of Multiple Columns or ValueNames, each with it's own sorting style specified by **ItemSortingStyle** property.

### Parameters

Name	Description
ParentIndex	Index value pointing to parent of the items to be sorted. To sort roots, set ParentIndex to -1.
SortPriority	(default 0) – identifies sort priority for the specified ValueName Multiple sorting styles, sorting keys and sorting methods may be specified. TList will sort data first according to specification of ItemSortingMethod, ItemSortingKey, and ItemSortingStyle having parameter SortPriority = 0, followed the sort criteria data having SortPriority 1, 2, 3, ...

### Sorted property settings:

Setting	Constant	Description
0	TLSORTSTATE_OFF	(Default) Sort OFF – Sorting is off. Additions or modifications to data in TList will not affect the order of the list, nor will changes to the sort properties.
1	TLSORTSTATE_ON	Sort On – Sorting is On. When first set TList performs an immediate sorting using current sort settings. Any changes to data or to sorting properties will immediately be reflected in a new sorting order.
2	TLSORTSTATE_RESET	Reset – Reset the sorting state – all sorting settings for a specific grid, node or LevelDef will be removed, and <b>Sorted</b> will be automatically set back to TLSORTSTATE_OFF

### SortingMethod property settings:

Setting	Constant	Description
0	TLSORT_ASCENDING	(Default) Sort items in the ascending order.
1	TLSORT_DESCENDING	Sort items in the descending order.

### SortingKey property settings:

The **SortingKey** property is a Variant which should be either Empty, a valid numeric column index, or a valid ValueName.

If SortingKey is Empty, sorting will be based on the default column of TList ( the column holding the main Tree / List data ).

If SortingKey contains a valid ValueName or numeric column index, sorting will be based on the ItemValues referenced by the ValueName or held in the Grid Column ( Grid.Coldef object ) pointed by column index.

SortingKey may be set to any Value Name for which ItemValues have been assigned to TList.

If SortingKey contains the special value "\_\_ObsoleteValue", TList will sort by the data held in the old ItemXXXvalue properties (ItemStrValue, ItemIntValue...).

---

Sorting may be performed on multiple columns in prioritized fashion by specifying multiple SortingKeys, each with a SortPriority parameter.

---

### SortingStyle property settings

The **SortingStyle** property should be set as the sum of the desired Bit Flag values from the following table:

Constant	Value	Description
TL_SORT_IGNORE_CASE	&H1	If set, ignore case while sorting.
TL_SORT_NUMBERS_SORT	&H2	If set, treat string representation of numbers as numbers. "10" comes after "2"
TL_SORT_STRING_SORT	&H4	If set, treat punctuation the same as symbols
TL_SORT_GROUP_ATTOP	&H8	If set, TList groups all items which have children at the top of the sorted sub-tree. <b>Note:</b> This Flag is processed by TList only for settings of SortingStyle with 2nd parameter (SortPriority ) = 0
TL_SORT_GROUP_ATBOTTOM	&H10	If set, TList groups all items which have children at the bottom of the sorted branch. <b>Note:</b> This Flag is processed by TList only for settings of SortingStyle with 2nd parameter (SortPriority ) = 0
TL_SORT_IGNORE_KANATYPE	&H20	Do not differentiate between Hiragana and Katakana characters. Corresponding Hiragana and Katakana characters compare as equal.
TL_SORT_IGNORE_NONSPACE	&H40	Ignore non-spacing characters
TL_SORT_IGNORE_SYMBOLS	&H80	Ignore symbols.
TL_SORT_IGNORE_WIDTH	&H100	Do not differentiate between a single-byte character and the same character

		as a double-byte character.
TL_SORT_FLOAT_NUMS_SORT	&H200	Apply for sorting of Floating point numbers. Takes fractional part of number into account when sorting.

**Note:** Bit Flag values, TL\_SORT\_GROUP\_ATTOP and TL\_SORT\_GROUP\_ATBOTTOM may not be applied together.

## Remarks

- Changing any of the SortXXX properties while the Sorted property is set to TLSORTSTATE\_ON will resort the data in TList.
- If only one column or ItemValue is used for sorting, the SortPriority parameter of the **SortingMethod**, **SortingStyle** and **SortingKey** properties may be omitted by users of Visual Basic but must be explicitly set by users of Visual C.
- When the Sorted property is applied to the a Tree Grid ( TList.Grid.Sorted = ) only rows which hold items of zero indentation level will be sorted. To sort other levels of hierarchy use ItemSortXXX properties (for sorting children of some node) or SortXXX properties of the Leveldefs object.

## Data Type

**Sorted** – Enum

**SortingKey** - Variant

**SortingMethod** – Integer

**SortingStyle** - Integer

## Example

### 1. Specify sorting within a grid

```
' **** Note if grid is heirarchic this will not sort chld items
' Sort – 1st by Last Name, 2nd by First Name – Ascending order, ignoring case

With TList1.Grid
  .Sorted = TLSORTSTATE_RESET
  .SortingKey(0) = "Last Name"
  .SortingMethod(0) = TLSORT_ASCENDING
  .SortingStyle(0) = TL_SORT_IGNORE_CASE

  .SortingKey(1) = "First Name"
  .SortingMethod(1) = TLSORT_ASCENDING
  .SortingStyle(1) = TL_SORT_IGNORE_CASE

  .Sorted = TLSORTSTATE_ON
End With
```

### 2. Sort An Item Grid by 3<sup>rd</sup> Column Numerically

```
With TList1.ItemGrid ( 27 ) ' item grid belonging to item index 27
  .Sorted = TLSORTSTATE_RESET
  .SortingKey(0) = 3
  .SortingMethod(0) = TLSORT_ASCENDING
  .SortingStyle(0) = TL_SORT_NUMBERS_SORT
  .Sorted = TLSORTSTATE_ON
End With
```

### 3. Specify Sorting By Heirarchic Level

```
' Sort all Root items by visible text
' – Ignore Case, keep items with children at the top
```

```

With TList1.LevelDefs(0)
    .Sorted = TLSORTSTATE_RESET
    .SortingKey = ""
    .SortingMethod = TLSORT_ASCENDING
    .SortingStyle = TL_SORT_IGNORE_CASE + TL_SORT_GROUP_ATTOP
    .Sorted = TLSORTSTATE_ON
End With

```

### [See Also](#)

**ItemSorted** property, **ItemSortingStyle** property, **ItemSortingMethod** property, **ItemSortingKey** property

---

## Spin Property (TListEditInfo Object)

### [Applies to](#)

TListEditInfo object

### [Description](#)

This property returns the **TListSpin** object controlling the editing with spin type of the editor for a data cell or cells

Note: The EditInfo.**Style** property must be set to TLEDITINFO\_SPIN before accessing the Spin property

### [Syntax](#)

```
[TListCellDef].EditInfo.Spin
```

### [Data Type](#)

TListSpin object

---

## Start Method

### [Applies To](#)

TListReport Object

### [Description](#)

The **Start** method is used to initialize the printing process. It takes the range of pages to be printed. It is allowed to call this method as many times as needed. If you need to print just some of the pages you should use the page numbers as provided by the **Report.Pages** collection. This collection is ready after the **PrepareForPrinting** method has been called. While the printing is in progress, TList generates the **BeginPage** and the **EndPage** events that can be used to print an additional information on every page (if needed).

### [Syntax](#)

```
[form.]TList.Report.Start([StartPage As Variant], [EndPage As Variant])
```

### [Remarks](#)

The **Start** method syntax has these parts:

Part	Description
<i>StartPage</i>	The index of the first page in the range to be printed
<i>EndPage</i>	The index of the last page in the range to be printed

---

## Step Property (TListSpin Object)

### [Applies to](#)



**TListCellDef** object

### Description

This property sets the **TListSpin** edit object increment value - the amount by which the value of the cell is changed in response to the user clicking the spin buttons.

### Syntax

```
[TListCellDef].EditInfo.Spin.Step [= Variant]
```

### Data Type

Variant

### Remarks

If the value of the Step property is negative, the actions of the Increment and Decrement spin buttons is reversed.

---

## Style Property (TListEditInfo Object)

### Applies to

**TListEditInfo** object

### Description

This property specifies defines the presentation / editing style for a cell.

### Syntax

```
[TListCellDef].EditInfo.Style = [enum%]
```

### Property Settings

Constant	Setting	Description
	-1	Removes current editing object, it will release any objects that. TList's base level text editing ( but without textbox object) may still be used.
TLEDITINFO_TEXTBOX	0	Sets TextBox presentation style.
TLEDITINFO_CHECKBOX	1	Sets CheckBox presentation style.
TLEDITINFO_COMBOBOX	2	Sets ComboBox presentation style.
TLEDITINFO_SPIN	3	Sets Spin presentation style.
TLEDITINFO_DATE_TIME	4	Sets Date/Time presentation style.

### Example

```
Dim tlCell as TListCellDef
Set tlCell = TList1.Grid.Coldefs(1).CellDef.EditInfo.Style
tlCell.EditInfo.Style = TLEDITINFO_CHECKBOX
```

### Remarks

To reset the Editing style to none – set the EditInfo property to “Nothing”

Set TList1.Grid.Coldefs(1).CellDef.EditInfo = Nothing  
or  
TList1.Grid.Coldefs(1).CellDef.EditInfo.Style = -1

---

## States Property (TListCheckBox Object)

### Applies to

TListCellDef object

### Description

This property defines the number of states (2 or 3) which the CheckBox can take. A two state checkbox is either Checked or UnChecked. A three state checkbox may also be in a Grayed (mixed) state.

### Values

The **States** property settings are:

Constant	Setting	Description
TLCHK_2STATES	0	(Default) 2 states checkbox: Displays checked or unchecked status.
TLCHK_3STATES	1	3 states checkbox: Displays checked, unchecked and grayed status.

### Syntax

```
[TListCellDef].EditInfo.CheckBox.States [= enum%]
```

### Sample Code

```
Dim cBox = TListCheckBox  
Set cBox = tlCell.EditInfo.Checkbox  
cBox.States = TLCHK_3STATES  
OR  
TList.Grid.Coldefs(4).CellDef.EditInfo.CheckBox.States= TLCHK_3STATES
```

---

## Style Property (TListComboBox Object)

### Applies to

TListEditInfo object

### Description

This property specifies the operating style of the combobox object

### Values

The **Style** property settings are:

Constant	Value	Description
TLCOMBO_DROP_DOWN	0	<b>(Default Value)</b> A ComboBox with a text entry box and a drop-down list
TLCOMBO_SIMPLE	1	ComboBox with a static edit box

TLCOMBO_DROP_DOWN_LIST	2	ComboBox with a drop-down list without text entry box
------------------------	---	---

Note: If the text in an editable cell has been hidden using [TListCellDef].**TextAlignment** = TLTEXTALIGNMENT\_NOT\_VISIBLE in order to show only pictures, make sure to set [TListCombobox].**Style** = TLCOMBO\_DROP\_DOWN\_LIST to allow selection of another item from the drop-down list. Otherwise you won't be able to edit the combobox.

## Syntax

```
[TListEditInfo].EditInfo.ComboBox..Style [= enum%]
```

# TabIndex Property

## Applies to

TList object

## Description

This property is the position of the control within the tab sequence of controls on a given form.

## Syntax

```
[form.]TList.TabIndex [= numeric_expr]
```

## Remarks

The valid range is any integer from 0 to (n-1), where n is the number of controls on the form that have a TabIndex property. Assigning a TabIndex value of less than 0 generates an error. For more information, see the description of the **TabIndex** property in the *Microsoft Visual Basic Language Reference*.

## Data Type

Integer

# TabStop Property

## Applies to

TList object

## Description

This property determines whether the control's focus can be reached by tabbing from other controls.

## Syntax

```
[form.]TList.TabStop [= bool_expr]
```

## Remarks

The TabStop property settings are:

Setting	Description
True	(Default) Designates the control as a tab stop.
False	Bypasses the control when the user is tabbing, although the control still holds its place in the actual tab order, as determined by the TabIndex property.

For more information, see the description of the **DragMode** property in the *Microsoft Visual Basic Language Reference*.

## Data Type

Boolean

---

## TabStopDistance Property

### Applies to

TList object

### Description

This property is the spacing between Tab locations. Measured in the scale of the control's container.

### Syntax

```
[form.]TList.TabStopDistance [= Integer%]
```

### Remarks

TList supports the tab character, CHR\$(9) within the **List** property for displayed text. The **TabStopDistance** property sets the relative spacing between tab stops. If set to a value less than the screen's resolution, the tab will instead be displayed as a quad character.

---

Note To build a true Grid with sizable columns and optional gridlines, it is possible to instruct TList to interpret Tabs as column delimiters when calling the **AddItem** method. This is controlled by the **ColDelimiter** and **ConvertTabsToCols** properties. Data can also be added directly to a grid. For further information refer to the section *Using TList Grids*.

---

### See Also

**ConvertTabsToCols** property; **ColDelimiter** property

---

## Tag Property

### Applies To

TList control

TListCellDef object

### Description

When applied to **TList** control:

- **Tag** property holds a string to be associated with the TList control.

When applied to **TListCellDef** object:

- **Tag** property holds a Variant to be associated with the **TListCellDef** object.
- The **Tag** property is the **default property** for **TListCellDef** object.

### Syntax

```
[form.]TList.Tag [= string_expr$]  
[form.]TListCellDefObject.Tag [= Variant]
```

### Remarks

For more information, see the description of the **Tag** property in the *Microsoft Visual Basic On-Line Help*.

### Data Type

TList's Tag property - String

TListCellDef object's Tag property - Variant

---

## Text Property

### Applies To

TList control

TListCellDef object

TListNode object

### Description

When applied to **TList** control:

- The **Text** property is the **default property** for TList. It returns the text for the item pointed to by the **ListIndex** property. Run-time only.

When applied to **TListCellDef** and **TListNode** Objects:

- The **Text** property returns the text for the cell this object refers to. Run-time only.

### Syntax

```
[form.]TList.Text[=str$]
```

### Remarks

TList.**Text** is equivalent to TList.**List**(TList.**ListIndex**):

### Data Type

String

---

## TextBox Property (TListEditInfo Object)

### Applies to

**TListEditInfo** object

### Description

This property returns a TListTextBox object controlling textbox editing for a data cell or cells

Note: The EditInfo.**Style** property must be set to TLEDITINFO\_TEXTBOX before accessing the Textbox property

### Syntax

```
[TListCellDef].EditInfo.TextBox
```

### Data Type

**TListTextBox** object

---

## TitleHeight Property

### Applies to

**TList** object

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## TitlePicture Property

### Applies to

**TList** object

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## TitleText Property

### Applies to

**TList** object

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## TitleVisible Property

### Applies to

TList object

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## TitleWidth Property

### Applies to

TList object

### Description

This property is not supported anymore; use Grids to obtain the same functionality.

---

## TitlesResize Event

### Applies to



TList object

### Description

This event is triggered when end-user starts to change a column width or row height by dragging a grid line in the grid row titles column, or in the grid column titles row .

An additional parameter has been added to the event to identify the grid in which the change is being made ( in case there are multiple grids held within TList.

### Syntax

```
Sub TList1_TitlesResize( [Index As Integer,] ByVal objGrid As TListProLibCtl.TlistGrid,  
     ByVal RowOrColumnFlag As Integer, ByVal RowOrColumnIndex As Long,  
     ByVal Size As Long)
```

### Parameters

Parameter	Description
ObjGrid	Returns the Grid object in which a row or column is being resized
RowOrColumnFlag	Specifies whether a row or a column is being resized. = TLTITLESRESIZE_ROW if user is changing the height of a row = TLTITLESRESIZE_COL if user is changing the height of a column
RowOrColumnIndex	The index of the row or column
Size	New column width or row height in pixels.

### Remarks

---

Note: This event will be called BEFORE the GridCellClick or Click events when the user is resizing a row or column.

---

---

## TList Property (TListTreeView)

### Description

This property returns a reference to the TList object that is used by TListTreeView control internally.

Using this property allows access to underlying TList object properties.

---

Note: TList virtual functionality is not accessible from within TListTreeView control.

---

### Syntax

```
TList = TListTreeView.TList
```

### Example

```
TListTreeView.TList.AddItem "some Text"
```

---

## TListCopyBuffer Function

### Applies to

TList object

### Description

See **CopyBuffer** method description.

### Syntax

```
Function TListCopyBuffer (ByVal hTreeBuffer&) As Integer
```

---

## TListFind ... Functions

### Applies to

TList object

### Description

See **FindItem** and **FindValue** methods description.

### Syntax

```
Function TListFindItem (tltTree As TList,  
    ➤ ByVal FindWhat As String,  
    ➤ ByVal nFlags As Integer,  
    ➤ ByVal nFromIndex As Long,  
    ➤ ByVal nToIndex As Long) As Long  
  
Function TListFindValue (tltTree As TList,  
    ➤ FindWhat As Variant,  
    ➤ ByVal nFlags As Integer,  
    ➤ ByVal nFromIndex As Long,  
    ➤ ByVal nToIndex As Long,  
    ➤ ByVal ValueName As Variant) As Long
```

### Parameters

Parameter	Description
<i>tltTree</i>	Name of the TList control.

<i>FindWhat</i>	String or Variant data being sought.
<i>nFlags</i>	How to search the item, This parameter is composed as the sum of bit flags - see description in the following table.
<i>nFromIndex</i>	Specifies the first item in the range.
<i>nToIndex</i>	Specifies the last item in the range.
<i>ValueName</i>	Optional. If present specifies the ValueName of the data to be searched.

*nFlags* parameter Flags:

Constant	Value	Description
TL_FI_DONTUSECASE	&H1	if set, search is not case-sensitive.
TL_FI_RELAXED	&H2	if set, valid item may include FindWhat as a substring of the item's text; otherwise the match must be exact.
TL_FI_BACKDIR	H100	if set, TList searches backwards from the end of the range.
TL_FI_SELOONLY	&H200	if set, TList searches only among selected items.
TL_FI_STARTSWITH	&H4	if set TList searches items whose text starts with the <i>FindWhat</i> .

### Returns

Index of the first found item or -1 if item wasn't found.

---

## TListFreeBuffer Function

### Applies to

TList object

### Description

See FreeBuffer method description.

### Syntax

```
Sub TListFreeBuffer(ByVal hTreeBuffer As Long)
```

---

## TListGetItemByXY Function

### Applies to

TList object

### Description

See GetItemByXY method description.

### Syntax

```
Function TListGetItemByXY(tlTree As TList,  
    ➡ ByVal X As Integer, ByVal Y As Integer,  
    ➡ ByVal nType As Integer) As Long
```

---

## TListGetItemRect Function

### Applies to



TList object

#### Description

See **GetItemRect** method.

---

## TListIndexByBM Function

#### Applies to

TList object

#### Description

See the **IndexByBM** method description.

#### Declarations

```
Function TListIndexByBM(tlTree As TList, ByVal hTreeBookmark As Long) As Long
```

---

## TListIsClipboardFormatAvailable Function

#### Applies to

TList object

#### Description

This function works identically to the **IsClipboardAvailable** property, but does not require an instance of TList.

Not available at design time.

#### Syntax

```
Function TListIsClipboardFormatAvailable() As Integer
```

#### Example

```
If TListIsClipboardFormatAvailable then  
    TList_Dest.Clipboard(TList.ListIndex) = 4  
End If
```

---

## TListIsValidBM Function

#### Applies to

TList object

#### Description

See **IsValidBM** method description.

#### Syntax

```
Function TListIsValidBM(tlTree As TList, ByVal hTreeBookmark&) As Integer
```

---

## TListIsValidBuffer Function

#### Applies to

TList object

#### Description

See **IsValidBuffer** method.

## Syntax

```
Function TListIsValidBuffer(ByVal hTreeBuffer As Long) As Integer
```

---

# TListLoadBuffer Function

## Applies to

TList object

## Description

See **LoadBuffer** method description

## Syntax

```
Function TListLoadBuffer(hTreeBuffer&, ByVal nFile As Integer) As Integer
```

---

# TListPasteBuffer Function

## Applies to

TList object

## Description

See **PasteBuffer** method description.

## Syntax

```
Function TListPasteBuffer (hTreeBuffer As Long) As Integer
```

---

# TListSaveBuffer Function

## Applies to

TList object

## Description

See **SaveBuffer** method description.

## Syntax

```
Function TListSaveBuffer (ByVal hTreeBuffer As Long ,  
    ➔ ByVal nFile As Integer) As Integer
```

---

# TListTranslateIndex Function

## Applies to

TList object

## Description

See **TranslateIndex** method description.

## Syntax

```
Function TListTranslateIndex (tltTree As TList ,  
    ➔ ByVal nFromIndexMethod As Integer,  
    ➔ ByVal nToIndexMethod As Integer ,  
    ➔ ByVal nFromIndex As Long) As Long
```

---

## TLNode Property (TListView)

### Description

This property returns a reference to the corresponding **TLNode** object. User can use this object for accessing advanced TList functionality (sorting, grid etc).

### Syntax

```
[TListNode] = TListView1.Nodes("key").TLNode
```

### Example

Here is a sample illustrating how to turn on sorting functionality:

```
'getting reference to TLNode object
Dim TmpNode As TLNode
Set TmpNode = TListView1.Nodes("SomeItem").TLNode
'using TLNode object interface for sorting all subordinate items
TmpNode.SortingStyle = TL_B_SORT_IGNORE_CASE
TmpNode.SortingMethod = 0 'sort in the ascending order
```

---

## ToolTipsBackColor Property

### Applies to

**TList** object

### Description

This property specifies the color of the Tool Tip box background. This property affects the TList display only if **ToolTipsViewStyle** property is set to 1.

### Syntax

```
[form.]TList.ToolTipsBackColor [= Color%]
```

### Data Type

Long

---

## ToolTipsDelay Property

### Applies to

**TList** object

### Description

TList supports display of a ToolTips window which presents the full text of a cell which does not otherwise fit within the available TList window or the width/height specified for a TList cell. The ToolTips are displayed directly over the TList item or grid cell and contain the full content of that item or cell.

The **ToolTipsDelay** property specifies the delay that will take place before showing ToolTips in TList.

ToolTipsDelay is specified in milliseconds. Setting ToolTipsDelay to 0 forces TList to display ToolTips as soon as the mouse is moved over a TList item.a.

### Syntax

```
[form.]TList.ToolTipsDelay [= Long&]
```

### Data Type

Long

---

## ToolTipsForeColor Property

### Applies to

TList object

### Description

This property specifies the color of the text of Tool Tip box. This property affects the TList display only if **ToolTipsViewStyle** property is set to 1.

### Syntax

```
[form.]TList.ToolTipsForeColor [= Color%]
```

### Data Type

Long

---

## ToolTipsMode Property

### Applies to

TList object

### Description

The property specifies whether ToolTips are shown while user is moving the mouse over an item.

### Syntax

```
[form.]TList.ToolTipsMode [= enum% ]
```

### Remarks

The **ToolTipsMode** property settings are:

Name	Value	Description
TLTOOLTIPSMODE_DISABLE	0	(Default) Do not show tooltips.
TLTOOLTIPSMODE_ENABLE	1	In this mode tooltip window is visible when the mouse pointer is over partially visible grid cell/item or over the tooltips window itself (still visible even if cross that cell/item boundaries).
TLTOOLTIPSMODE_STANDARD	2	In this mode tooltip window is visible only when the mouse pointer is over partially visible grid cell/item and does not cross that cell/item boundaries. This behavior is similar to a standard behavior of Windows Explorer.

### Data Type

Integer

---

## ToolTipsText Property

### Applies to

TList object

### Description

Setting the ToolTipsText property defines a string to be shown as a tooltip whenever the mouse hangs for some time over the TList control itself.

## Syntax

```
[form.]TList.ToolTipText [= "some string"]
```

## Example

```
'display corresponding text as a tooltip  
TList1.ToolTipText = "Your mouse is over the TList control"
```

## Remarks

This property is not related to the tooltip to be shown over a list or tree item, or a grid cell. This tooltip is not controlled by the ToolTipsMode or ToolTipsDelay property and not related to any other Tooltips properties – it is disabled only by setting the property to an empty string.

## Data Type

String

---

# ToolTipViewStyle Property

## Applies to

TList object

## Description

This property specifies the colors used to paint ToolTips.

## Syntax

```
[form.]TList.ToolTipViewStyle [= enum% ]
```

## Remarks

The **ToolTipViewStyle** property settings are:

Setting	Description
0 - Predefined Colors	(Default) TList's item color is used to draw ToolTips.
1 - User-defined colors	<b>ToolTipBackColor</b> and <b>ToolTipForeColor</b> properties specify the colors which are used to draw ToolTips.

## Data Type

Integer

---

# Top Property

## Applies to

TList object

## Description

This property determines the vertical distance between the top edge of TList control and its container.

## Syntax

```
[form.]TList.Top [= numeric_expr ]
```

## Remarks

Setting of this property updates the control unless the **Redraw** property is set to False. For more information, see the description of the **Top** property in the *Microsoft Visual Basic On-Line Help*.

## Data Type

Single

---

## TopIndex Property

### Applies to

TList object

### Description

This property sets and returns the item that appears in the topmost position in the TList control. If the specified item is not visible because its parent is collapsed, the next visible item will be set. The default is 0, or the first item.

Not available at design time.

### Syntax

```
[form.]TList.TopIndex[=top&]
```

### Remarks

Setting of **TopIndex** property updates the control unless the **Redraw** property is set to False. The **TopIndex** property may be used to scroll the control vertically.

### Data Type

Long

---

## TranslateIndex Method

### Applies to

TList object

### Description

The **TranslateIndex** method may be used to translate an index from one index method to another. Not available at design time.

### Syntax

```
[form.]TList.TranslateIndex (tltTree As TList ,  
    ➡ ByVal nFromIndexMethod As Integer,  
    ➡ ByVal nToIndexMethod As Integer ,  
    ➡ ByVal nFromIndex As Long) As Long
```

### Example

```
' translate an index from the current indexation  
' method to TL_SYSLEVEL index methods  
Dim DestIndex&, SourceIndex&  
...  
SourceIndex& = TList1.ListIndex  
...  
DestIndex& = TList1.TranslateIndex(TList1.CurrentIndexMethod, TL_SYSLEVEL, SourceIndex&)
```

---

## TransparentBackground Property

### Applies to

TList object

### Description

Setting the **TransparentBackground** property determines whether TList background is transparent.

430 • Properties, events, methods, functions reference  
Control

Programmer's Guide TList 8 ActiveX

### Syntax

```
[form.]TList.TransparentBackground [= bool%]
```

### Remarks

There may be certain controls which do not show through a transparent TList. But Visual Basic forms and all graphic controls function properly with TList in transparent mode.

### Data Type

Boolean

### See Also

**UpdateBackground** method

---

## TransparentBitmap Property

### Applies to

TList object

### Description

Setting the **TransparentBitmap** property determines whether TList displays bitmaps as transparent. If this property is set to True the **TransparentBitmapColor** property determines the transparent color.

### Syntax

```
[form.]TList.TransparentBitmap [= bool%]
```

### Data Type

Boolean

### See Also

**TransparentBitmapColor** property

---

## TransparentBitmapColor Property

### Applies to

TList object

### Description

Determines the color, which will be considered transparent as bitmaps are displayed. These property settings are used only if the **TransparentBitmap** property is set to True.

### Syntax

```
[form.]TList.TransparentBitmapColor [= Color&]
```

### Data Type

Long

### See Also

**TransparentBitmap** property

---

## TreeGrid Property

### Applies To

TListGrid object

## Description

This property determines whether a grid is a *Tree Grid* or an *Item Grid*. Read-only. Returns True if this is *Tree Grid* and False otherwise.

## Syntax

```
TListGridObject.TreeGrid [= bool%]  
[form.]TList.Grid. TreeGrid [= bool%]  
[form.]TList.ItemGrid(ItemIndex&). TreeGrid [= bool%]
```

## Data Type

Boolean

---

# TreeLinesColor Property

## Applies to

TList object

## Description

Setting the **TreeLinesColor** property determines what color is used to draw tree lines.

## Syntax

```
[form.]TList.TreeLinesColor [= Color&]
```

## Remarks

Setting of **TreeLinesColor** property updates control display unless the **Redraw** property is set to False.

## Data Type

Long

---

# TreeLinesStyle Property

## Applies to

TList object

## Description

Setting the **TreeLinesStyle** property determines how tree lines are drawn.

## Syntax

```
[form.]TList.TreeLinesStyle [= enum%]
```

## Settings

The TreeLinesStyle property setting are: Constant	Setting	Description
TLTREEL_SOLID	0	(Default) Solid lines.
TLTREEL_DASH	1	Dash lines.
TLTREEL_DOT	2	Dotted lines.
TLTREEL_DASHDOT	3	Dash-dot lines
TLTREEL_DASHDOTDOT	4	Dash-dot-dot lines
TLTREEL_3DINSET_2COLOR	5	3D-Inset using 2 colors. TList uses Windows settings for specification of shadow and highlight color for 3D tree lines.



TLTREEL_3DRAISED_2COLOR	6	3D-Raised using 2 colors. TList uses Windows settings for specification of shadow and highlight color for 3D tree lines.
TLTREEL_3DINSET_3COLOR	7	3D-Inset using 3 colors. TList uses Windows settings to for specification of shadow and highlight color for 3D tree lines.
TLTREEL_3DRAISED_3COLOR	8	3D-Raised using 3 colors. TList uses Windows settings for specification of shadow and highlight color for 3D tree lines.
TLTREEL_3D_USER_COLORS	9	3D-User-defined Shadow and highlight colors specified by <b>TreeLinesHighlightColor</b> and <b>TreeLinesShadowColor</b> properties
TLTREEL_3D_AUTO_COLORS	10	3D-Calculated Colors. TList automatically calculates <b>TreeLinesHighlightColor</b> and <b>TreeLinesShadowColor</b> colors based on current <b>TreeLinesColor</b> property.

---

## TreeLinesColor, TreeLinesHighlightColor and TreeLinesShadowColor properties

### Applies to

TList object

### Description

- **TreeLinesColor** property determines the base color used to draw tree lines within TList.
- **TreeLinesHighlightColor** and **TreeLinesShadowColor** properties specify highlight and shadow colors used for 3-D tree lines when **TreeLinesStyle** property is set to TLTREEL\_3D\_USER\_COLORS.
- **TreeLinesHighlightColor** and **TreeLinesShadowColor** properties return values dependant on **TreeLinesStyle** property.

### Syntax

```
TList.TreeLinesColor = [color]
TList.TreeLinesHighlightColor = [color]
TList.TreeLinesShadowColor = [color]
```

### Data Type

Color / Long

### Remarks

Setting of TreeLinesColor property updates control display unless the Redraw property is set to False.

Setting either **TreeLinesHighlightColor** or **TreeLinesShadowColor** automatically sets the **TreeLinesStyle** property to `TLTREEL_3D_USER_COLORS`.

Note: Either **TreeLinesHighlightColor** or **TreeLinesShadowColor** can be set to transparent. This allows drawing 3D **TreeLines** with 2 colors.

### See Also

**TreeLinesStyle** property

---

## TriggerEvents Property

### Applies to

**TList** object

### Description

The **TriggerEvents** property determines whether collapse and expand events will be generated when an item and its children are collapsed or expanded.

### Syntax

`[form.]TList.TriggerEvents [= enum%]`

### Settings

**NOTE** The property's behavior was changed since **TList 5.0**. Prior to **TList 6.0**, programmers could control only the firing the collapse events; starting with **TList 6.0** full control is provided over firing of the expand event as well.. To turn on new functionality it is necessary to set the **TL\_MOD\_EXPAND\_EVENTS** flag in the **Modifications** property. If the modification's flag is not specified the property will work as in **TList 5**.

Below you can see two tables describing settings for old and new behavior.

New **TriggerEvents** property setting

(with **TL\_MOD\_EXPAND\_EVENTS** flag set in **Modifications** property):

Setting	Description
0	Do not generate any events.
1	Generate only <b>Collapse</b> events for all collapsed items.
2	Generate only <b>Expand</b> events for all collapsed items.
3	Generate both <b>Expand/Collapse</b> events for all collapsed items.

Old **TriggerEvents** property setting are:

(with **TL\_MOD\_EXPAND\_EVENTS** flag cleared in **Modifications** property):

Setting	Description
0	(Default) Do not generate <b>Collapse</b> event for children. Expand and Collapse events for the item itself are generated.
1	Generate <b>Collapse</b> event for children. Expand and Collapse events for the item itself are generated.

### Data Type

Integer

---

## UpdateBackground Method

### Applies to

TList object

### Description

To optimize repainting, TList saves an image of the background in its internal buffer. If for example, you changed the underlying form's background and TList's Transparent property is set to True, you have to update TList's background manually.

### Syntax

```
Sub TList.UpdateBackground
```

---

## UpdateImages Method (TListTreeView )

### Description

This method is called in order to notify **TListTreeView** control about changes inside an ImageList control. After any images within an ImageList control connected with **TListTreeView** are added, modified or removed, the UpdateImage method must be called to update the corresponding **TListTreeView** control.

### Syntax

```
TListTreeView.UpdateImages
```

---

## Url Property

### Applies To

TListCellDef object

### Description

TList's Web Auto Navigation feature uses relative URLs stored in the **Url** property to navigate to a web site when a user double-clicks on a cell.

Not available at design time.

### Syntax

```
[form.]TList.Grid.Cells(Row, Col).CellDef.Url [ = String$]
```

### Example

Below is a sample, which points to a very interesting web page:

```
TList1.ItemGrid(0).Cells(10,40).CellDef.Url(2)= _  
"http://www.bennet-tec.com/common/whoarewe.htm"
```

### Data Type

String

### See Also

WebAutoNavigate property

---

## Value Property

### Applies To

TListValue Object

TListGridCell Object

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

## Description

**TListView**'s **Value** property stores data of Variant type associated with this object. Default property.

**TListGridCell**'s **Value** property returns **TListView** object associated with this grid cell.

## Syntax

```
TListViewObject.Value [ = Variant]
[form.]TList1.ItemValues(ItemIndex&, ValueNameIndex).Value[ = Variant]
[form.]TList1.ItemGrid(ItemIndex&).Cells(Row&, Col&).Value.Value[ = Variant]
[form.]TList1.Grid.Cells(Row&, Col&).Value.Value[ = Variant]
```

---

# ValueName Property

## Applies To

**TListView** Object

**TListColDef** Object

## Description

The **ValueName** property is used to refer to named data which may be associated with each item in the list. This associated data may be hidden from the end-user or displayed within columns if a table/grid has been set up.

## Syntax

```
TListViewObject.ValueName
[form.]TList1.ItemValues(ItemIndex&, ValueNameIndex). ValueName
[form.]TList1.ItemGrid(ItemIndex&).Cells(Row&, Col&).Value.ValueName
[form.]TList1.Grid.Cells(Row&, Col&).Value.ValueName
```

## Remarks

A given item/row in **TList** may have associated data elements as defined by the **ItemValues** property.

```
TList1.ItemValues(25, "CityName").Value = "Paris"
TList1.ItemValues(25, "Name").Value = "Michael"
```

These two statements create 2 new associated data elements with ValueNames, "CityName" and "Name", and associate these values with the 25<sup>th</sup> item.

The **ValueNames** property can be used to identify the names of associated data elements:

```
Dim X As TListView
For Each X In TList1.ItemValues(100)
    Print X.ValueName
Next
```

Setting the **ValueName** for a **ColDef** object display associated data values in a grid column. To show the city name values in the I+1<sup>th</sup> column of a grid use:

```
TList1.Grid.ColDefs(I).ValueName = "CityName"
```

## Data Type

String

---

# Value and OldValue Properties (TListEditingChangeInfo object)

## Applies to

## TListEditingChangeInfo object

### Description

The Value property defines the new value being entered into the edit object (**TListTextBox**, **TListComboBox**, **TListCheckBox**, etc). It can be modified inside the GridCellEditingChange, ItemEditingChange events to "correct" data entered by a user.

The OldValue property returns the value held by the edit object before it was changed by the end-user.

Read-only.

---

Note: when setting the Value property by code, the new value will be accepted only if the value is valid, otherwise the value setting will be ignored and the data entered by the user will be used. For example: When working with the TListCheckBox edit object the TListEditingChangeInfo.Value property can only be set to one of the values specified by [TListEditInfo].CheckBox.CheckedValue, [TListEditInfo].CheckBox.UncheckedValue, etc properties.

---

### Syntax

[TListEditingChangeInfo].Value = [Variant]

### Example

```
' DATA VERIFICATION
Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _
    ByVal objGridCell As TListGridCell, _
    ByVal objChangeInfo As TListEditingChangeInfo)

    ' Verify that data being entered in column 3 is a valid number

    If GridCell.Col = 3 Then
        'check the text and if it is not a valid number restore previous text
        If IsNumeric(objChangeInfo.Value) = False Then
            ObjChangeInfo.Value = objChangeInfo.OldValue 'restore data
        End If
    End If
End Sub
```

### See Also

**GridCellEditingChange** and **ItemEditingChange** events, **ValueType**, **Value**, **OldValue**, **SelStart**, **SelLength**, **EditInfoObject** properties

---

## ValueType Property (TListEditingChangeInfo object)

### Applies to

TListEditingChangeInfo object

### Description

This property identifies which value is being changed within the edit object (**TListTextBox**, **TListComboBox**, **TListCheckBox**, etc).

Property is read-only.

### Syntax

[TListEditingChangeInfo].ValueType = [enum%]

### Property Settings

Constant	Setting	Description
----------	---------	-------------

Programmer's Guide TList 8 ActiveX ControlProperties, events, methods, functions reference •

9		
TLED_CHANGE_TEXTBOX	0	Value contains the text that was entered by user, OldValue contains the previous data.
TLED_CHANGE_CHECKBOX	1	Value contains the new data for <b>TListCheckBox</b> object (one of the values specified by [TListEditInfo].CheckBox.CheckedValue, [TListEditInfo].CheckBox.UncheckedValue, [TListEditInfo].CheckBox.GrayedValue. OldValue identifies the previous checkbox value.
TLED_CHANGE_COMBOBOX_EDITAREA	2	Value returns the text being entered by user in the edit area of TListComboBox object. OldValue contains the previous data.
TLED_CHANGE_SPIN	3	Value returns the value of the TListSpinBox object. OldValue contains the previous data.
TLED_CHANGE_DATETIME	4	Value returns the value of the TListDateTime object. OldValue contains the previous data.
TLED_CHANGE_COMBOBOX_LISTINDEX	5	Value returns the index of the item of the combobox list item that was selected by mouse or keyboard. OldValue contains the previous index.

### Example

```

Private Sub TList1_GridCellEditingChange(ByVal ItemIndex As Long, _
    ByVal objGridCell As TListGridCell, _
    ByVal objChangeInfo As TListEditingChangeInfo)

    Select Case objChangeInfo.ValueType
        Case TLED_CHANGE_TEXTBOX
            Text1.Text = "Text Editing Change " & Value
        Case TLED_CHANGE_CHECKBOX
            Text1.Text = "Checkbox Editing Change: " & Value
        Case TLED_CHANGE_COMBOBOX_EDITAREA
            Text1.Text = "ComboBox Editing Text Area Change: " & Value
        Case TLED_CHANGE_SPIN
            Text1.Text = "Spin Editing Change: " & Value
        Case TLED_CHANGE_DATETIME
            Text1.Text = "Date/Time Editing Change: " & Value
        Case TLED_CHANGE_COMBOBOX_LISTINDEX
            'get a reference to the combobox item that was selected
            Dim objComboltem as TListComboltem
            Set objComboltem = _
                objChangeInfo.EditInfoObject.ComboBox.Items(objChangeInfo.Value)
            'update some outside TextBox object depending on the item index
            Text1.Text = "Editing Listbox Selection Change: " & _
                objComboltem.CellValue
            Text1.BackColor = objComboltem.BackColor
    End Select

```

End Sub

### See Also

**GridCellEditingChange** and **ItemEditingChange** events, **ValueType**, **Value**, **OldValue**, **SelStart**, **SelLength**, **EditInfoObject** properties

---

## Version Property

### Applies to

**TList** object

### Description

This property returns the current version for the control. Read-only at run-time.

This property returns major version, minor version, and build number, for example: if TList has major version 4, minor version 5 and build 25, Version would return decimal 40525. That is, the following mathematical formula is used:

Version = MajorVersion \* 10000 + MinorVersion \* 1000 + BuildNumber.

### Syntax

[form.]TList.Version

### Data Type

Long

---

## ViewStyle Property

### Applies to

**TList** object

### Description

This property determines the way an item will be displayed, ie: which visual elements to include in the display.

### Syntax

[form.]TList.ViewStyle [= setting%]

### Settings

The **ViewStyle** property settings are:

Setting	Description (which visual elements are displayed)
0	(Default) Picture, Tree lines and Text associated with the item .
1	Picture and Text
2	Picture and Tree lines
3	Picture
4	Text and Tree lines
5	Text

### Remarks

Setting of **ViewStyle** property updates the control unless the **Redraw** property is set to False. See **ViewStyleEx** property for extended settings.

## Data Type

Integer

---

# ViewStyleEx Property

## Applies to

TList object

## Description

The **ViewStyleEx** property extends the settings of **ViewStyle** providing for control over display of Plus/Minus pictures and Mark Pictures.

## Syntax

```
[form.]TList.ViewStyleEx[ = enum%]
```

## Settings

The **ViewStyleEx** property settings are:

Setting	Description
0	(Default) Don't display either the Plus/Minus image or Mark Pictures.
1	Display Plus/Minus image.
2	Display Mark pictures and Plus/Minus image.
3	Display Mark pictures.

## Data Type

Integer

---

# Visible Property

## Applies To

TList control

TListCellDef object

TListGrid object

## Description

TList's **Visible** property determines whether the TList control is visible or hidden.

TListCellDef object's **Visible** property determines whether the column header is visible or hidden.

TListGrid object's **Visible** property determines whether the grid is visible or hidden.

## Syntax

```
[form.]TList.Visible [= bool_expr]  
[form.]TList.ColDefs(ColDefIndex&).Visible [= bool_expr]  
[form.]TList.Grid.Visible [= bool_expr]  
[form.]TList.ItemGrid(ItemIndex&).Visible [= bool_expr]
```

## Remarks

The **Visible** property settings are:

Setting	Description
True	(Default) Control/Column Header/Grid is visible.
False	Control/Column Header/Grid is hidden.



For more information, see the description of the **Visible** property in the *Microsoft Visual Basic Language Reference*.

#### Data Type

Boolean

---

## Visible Property (TListNode Object)

#### Applies To

TListNode object

#### Description

TListNode Object's **Visible** property determines whether the item is visible or hidden. Read-only.

#### Syntax

```
TListNode.Visible [= bool_expr]
```

---

## VisualRoot Property

#### Applies to

TList object

#### Description

This property determines which portion of the tree to display. It specifies the index of an item whose children should be displayed. If set to -1, the whole tree is displayed.

#### Syntax

```
[form.]TList.VisualRoot [= Long]
```

#### Remarks

For more information, see *How to Use the VisualRoot Property* chapter.

#### Data Type

Long

---

## WebAutoNavigate Property

#### Applies to

TList object

#### Description

This property turns on or off TList's Web Auto Navigate feature. If this feature is turned on, TList navigates to a Web document whose URL is specified by the **ItemUrl** property of a double clicked item or by **Url** property of a double clicked cell.

To determine the URL for Web navigation, TList appends the contents of **ItemURL** or **Url** property to the value of the **WebURLBase** property.

The **WebTargetFrame** property is used to specify a name of the frame in which to display the loaded Web document.

#### Syntax

```
[form.]TList.WebAutoNavigate [= enum% ]
```

#### Remarks

The **WebAutoNavigate** property settings are:

Programmer's Guide TList 8 ActiveX Control Properties, events, methods, functions reference •

Setting	Description
0	(Default) TList Web Auto Navigate feature is turned off.
1	TList Web Auto Navigate feature is turned on. Navigate on double click, use <b>ItemURL/Url</b> property (appended to <b>WebURLBase</b> ) as an URL.

#### Data Type

Integer

---

## WebTargetFrame Property

#### Applies to

TList object

#### Description

**WebTargetFrame** property is used to specify a name of the frame in which to display the loaded Web document.

#### Syntax

```
[form.]TList.WebTargetFrame [ = string$ ]
```

#### Data Type

String

---

## WebURLBase Property

#### Description

To get a full URL, which must be used for document location, TList adds the contents of **ItemURL** property to the value of the **WebURLBase** property.

#### Syntax

```
[form.]TList.WebURLBase [ = string$ ]
```

#### Data Type

String

---

## WebGoBack Method

#### Applies to

TList object

#### Description

Navigates to the previous item in the history list.

#### Syntax

```
[form.]TList.WebGoBack
```

---

## WebGoForward Method

#### Applies to

TList object

#### Description

This method navigates to the next item in the history list.

### Syntax

```
[form.]TList.WebGoForward
```

---

## WebNavigate Method

### Applies to

TList object

### Description

Calling this method instructs the web browser to navigate to a document specified by a URL.

### Syntax

```
[form.]TList.WebNavigate(ByVal URL As String, ByVal TargetFrameName As String,  
➡ ByVal Flags As Integer )
```

### Remarks

The **WebNavigate** method has these parts:

Part	Description									
URL	A string expression that evaluates to the URL of the resource that the browser is to display.									
TargetFrameName	A string expression specifying the name of a frame in which to display the resource. This string can be empty.									
Flags	<p>A constant or value that specifies whether to display the resource in a new window. It is one of these values:</p> <table><tr><th><u>Constant</u></th><th><u>Value</u></th><th><u>Meaning</u></th></tr><tr><td>tlWebNavigateCurWindow</td><td>0</td><td><b>(Default)</b> Open in the current window.</td></tr><tr><td>tlWebNavigateNewWindow</td><td>1</td><td>Open in a new window.</td></tr></table>	<u>Constant</u>	<u>Value</u>	<u>Meaning</u>	tlWebNavigateCurWindow	0	<b>(Default)</b> Open in the current window.	tlWebNavigateNewWindow	1	Open in a new window.
<u>Constant</u>	<u>Value</u>	<u>Meaning</u>								
tlWebNavigateCurWindow	0	<b>(Default)</b> Open in the current window.								
tlWebNavigateNewWindow	1	Open in a new window.								

---

## WheelScrolling Property

### Applies to

TList object

### Description

Provides the developer with control over IntelliMouse functionality of TList control. It is possible to use either the default scrolling behavior or to handle the behavior manually by trapping the corresponding **MouseWheel** event.

### Syntax

```
TList.WheelScrolling[ = Flags&]
```

### Data Type

Long

### Settings

Constant	Value	Description
TL_WHEELSCRL_NONE	0	TList takes no automatic actions.
TL_WHEELSCRL_ITEM	&H1	TList scrolls a tree item by item if user rotates the mouse wheel.

TL_WHEELSCRL_PAGE	&H2	TList scrolls a tree page by page if user rotates the mouse wheel and SHIFT key is held down.
TL_WHEELSCRL_ROOT	&H4	TList scrolls a tree root by root if user rotates the mouse wheel and CTRL key is held down.
TL_WHEELSCRL_LISTINDEX	&H8	TList automatically updates the value of ListIndex property to insure that the item pointed to by ListIndex is always inside the visible part of the tree while wheel scrolling operations are performed.
TL_WHEELSCRL_SYSTEM	&H16	When user rotates the mouse wheel TList scrolls a tree using Windows system settings for wheel scrolling (by default: 3 items at once). Note: this flag is ignored if user specifies TL_WHEELSCRL_ITEM flag.

---

Note: With TL\_WHEELSCRL\_LISTINDEX set, If the item pointed to by ListIndex is scrolled above the client area, then ListIndex is reset to the value of TopIndex. If the item pointed to by ListIndex is scrolled below the client area then ListIndex is reset to the value of BottomIndex.

---

### See Also

**MouseWheel** event

---

## Width Property

### Applies To

**TList** control

**TListColDef** object

### Description

TList's **Width** property contains the width of the TList control.

A **TListColDef** object's **Width** property determines the width of a column in twips.

### Syntax

```
[form.]TList.Width [= xsize!]  
[form.]TList.Grid.ColDefs(Col&).Width [= xsize!]  
[form.]TList.ItemGrid(ItemIndex&).ColDefs(Col&).Width [= xsize!]
```

### Remarks

**TList's Width** property: for more information, see the description of the **Width** property in the *Microsoft Visual Basic Language Reference*.

**TListColDef** object's **Width** property:

You can use this property to set the width of any column at run time.

Use **TListColDef** object's **Visible** property to create invisible columns.

The following code allows an end user to resize columns and then displays the width of column (0) in a textbox.

```
Sub Form1_Load()  
    TList1.Grid.AllowResizing = TLRESIZING_COLS_AND_ROWS_AND_TREEGRIDCAPTION  
End Sub  
  
Sub TList1_MouseUp (Button As Integer, Shift As _  
    Integer, X As Single, Y As Single)  
    MsgBox "Width = " TList1.ActiveGrid.ColDefs(0).Width  
End Sub
```

## Data Type

Single

---

# WidthOfText Property

## Applies to

TList object

## Description

This property specifies the wrapping width of text for an item which can display multiple lines of text. An item can accept multiple lines of text only if the **DefMultiLine** property is set to True or the **ItemMultiLine** property for this item is set to True. See *Settings* below for more detailed description.

## Syntax

[form.]TList.WidthOfText[=xsize!]

## Settings

The **WidthOfText** property settings are:

Setting	Description
-1	Multi-line text items are wrapped to fit within TList's client area. The minimum wrap width is determined by <b>WidthOfTextMin</b> property
0	(Default)Multi-line text items are wrapped to fit within TList's client area. But the width of the multiple-line text for each item is calculated as if the item is of zero indentation regardless of its real indentation.
> 0	Text is wrapped to the specified length as Measured in terms of the units of TList's container.

## Remarks

Setting of **WidthOfText** property updates the control unless the **Redraw** property is set to False. This property has no affect on items whose **ItemMultiLine** property is False unless **DefMultiLine** is True.

## Data Type

Single

---

# WidthOfTextMin Property

## Applies to

TList object

## Description

This property specifies the minimum wrapping width of text for an item which can display multiple lines of text. An item can accept multiple lines of text only if the **DefMultiLine** property is set to True or the **ItemMultiLine** property for this item is set to True. See Setting below for more detailed description. Used only if **WidthOfText** is set to -1.

## Syntax

[form.]TList.WidthOfTextMin[=xsize!]
--------------------------------------

## Settings

Setting of **WidthOfTextMin** property updates the control unless the **Redraw** property is set to False. This property has no affect on items whose **ItemMultiLine** property is False unless **DefMultiLine** is True.

#### Data Type

Single

---

## XOffset Property

#### Applies to

TList object

#### Description

This property sets the left offset of TList elements with indent = 0 from the left edge of the control. Measured in terms of the units of TList's container.

#### Syntax

```
[form.]TList.XOffset [=xsize!]
```

#### Data Type

Single

---

## Zoom Property

#### Applies To

TListReport object

#### Description

This property specifies a zoom factor in percents (from 1% to 10000%) to be used when printing from TList.

Not available at design-time.

#### Syntax

```
TListReportObject.Zoom [ = zoom%]  
[form.]TList.Report.Zoom [ = zoom%]
```

#### Data Type

Integer

---

## ZOrder Method

#### Applies to

TList object

#### Description

Places a control at the front or back of the z-order within its graphical level.

#### Syntax

```
[form.]TList.ZOrder
```

#### Remarks

For more information, see the description of the **ZOrder** method in the *Microsoft Visual Basic Language Reference*

# Error messages

## Trappable Errors

The following table lists the trappable errors for this control.

Error number	Message explanation
20101	TLERR_BADINDENTATION Bad indentation. The nesting level of an item cannot exceed 255.
20102	TLERR_WRONGPARAMS Wrong parameters This error occurs when any attempt is made to assign an invalid string to the <b>CurrentParent</b> (or obsolete <b>CurrentItem</b> ) property.
20103	TLERR_ITEMNOTFOUND Item not found This error occurs when any attempt is made to pass the <b>CurrentParent</b> (or obsolete <b>CurrentItem</b> ) property a string referencing a non-existing item.
20104	TLERR_PARENTNOTEXPANDED Parent not expanded An item must be visible (expanded) in order to expand its subordinate items.
20105	TLERR_INITFAULT Initialization error. Error during control initialization. Usually, this indicates insufficient memory.
20106	TLERR_CANNOTSHOW Can not show an item There is enough memory to store new items, but no room to show them. It is possible in this situation to collapse some items and then try to add or expand items again.
20107	TLERR_INVALIDSEPARATOR Invalid path separator Path separators can not be empty strings or strings that start with a period ('.').
20108	TLERR_INVALIDTREEBUFFER Invalid <i>tree buffer</i> The long number being used as a pointer to a <i>tree buffer</i> is invalid. Use only values returned by one of the Copy properties.
20109	TLERR_BIGPICSIZE The Picture Size is too large.

This error occurs when any attempt is made to assign an invalid picture size to the **ItemImageDefHeight** or **ItemImageDefWidth** properties or when any attempt is made to assign an invalid shift step size to the **ShiftStep** property. See Limitations for more details.

- 20110 TLERR\_NOTHINGTOCOPY  
Nothing to copy  
An attempt was made to use property **CopySelected** when no items were selected.  
– OR –  
An attempt was made to use **CopyItemSub** on an item having no subordinates.
- 20111 TLERR\_TOMANYITEMS  
Too many items in the list  
An attempt was made to add more items to the list than control can accept. See Limitations for more details.
- 20112 TLERR\_TOMANYSUBITEMS  
Too many subordinate items in one item  
An attempt was made to add more subordinate items to one of the items in the list than the control can accept. See Limitations for more details.
- 20123 ERR\_TL\_REPORT\_INVALIDMARGINS  
Invalid margins were specified.
- 20124 ERR\_TL\_REPORT\_INVALIDDC  
An invalid DC was specified.
- 20125 ERR\_TL\_REPORT\_INVALIDPAGERANGE  
An invalid page range was specified as the parameter for the Start method
- 20126 ERR\_TL\_CANTCONNECTTOTLIST  
User tries to use CellDef object that has already been disconnected from TList object (for example: **corresponding** item was deleted from tree)
- 20127 ERR\_TL\_VERSION\_SERIALNUMBERLIMITATION  
User tries to use functionality that isn't available for used serial number.
- 20128 ERR\_TL\_AUTODRAG\_CANNOTINITIATEDRAG  
Auto drag/drop functionality is available and worked properly only under VB environment.
- 20129 ERR\_TL\_AUTODRAG\_CANNOTCONNECTSOURCE  
Source control isn't available in order to complete drag/drop operation.
- 20130 ERR\_TL\_AUTODRAG\_CANNOTTMOVE  
Can't perform specified moving operation.
- 20131 ERR\_TL\_AUTODRAG\_CANNOTTRANSFER  
Can't transfer data from source to destination tree.
- 20132 ERR\_TL\_CAN\_NOT\_BE\_SORT  
Virtual trees can't be sorted.
- 20134 ERR\_TL\_INVALIDEDITSTYLE  
The style of TListEditInfo object does not correspond to this editing object.
- 20135 ERR\_TL\_ITEM\_NOT\_FOUND  
Item not found
- 20136 ERR\_TL\_MIN\_BIGGER\_MAX  
Minimum must not exceed maximum



20137	ERR_TL_MAX_SMALLER_MIN Maximum must exceed minimum
20138	ERR_TL_INVALID_PROPERTY Invalid property name or ID
20139	ERR_TL_ROW_NOT_FOUND Row not found
20140	ERR_TL_ROW_NOT_VISIBLE Row is not visible or does not exist
20141	ERR_TL_COL_NOT_VISIBLE Column is not visible or not exist
20142	ERR_TL_INAPPLICABLE_IN_SINGLE_SEL_MODE Inapplicable call in single selection mode.
20143	ERR_TL_CANTCONNECTTOGRID Can't connect to grid.
20144	ERR_TL_DRAGMODESCONFLICT Drag modes conflict. Can't use both Auto drag and OLE drag at the same time
20145	ERR_TL_ITEM_NOT_VISIBLE Item is not visible or not exist
20146	ERR_TL_WRONG_SUB_SORT_KEY Invalid SortPriority index
20147	ERR_TL_CELL_CAN_NOT_BE_ACTIVATED Cell can not be activated
20148	ERR_TL_ROW_CAN_NOT_BE_ACTIVATED Row can not be activated

---

Note Symbolic constants for error code definitions can be found in the file TLIST8.BAS distributed with the control.

---



# Index

## A

AbortWindow property 171  
AbortWindowStyle property 171  
About property 171  
Activatable Property 172  
Activate Method (TListGrid object) 173  
ActivationMode Property (TListGrid object) 174  
ActiveCell Property 176  
ActiveGrid property 176  
ActiveRow Property 176  
Add method 178, 180, 181  
Add method (TListComboItems Object) 177  
Add method (TListSelectedGridCells Object) 178  
Add method (TListSelectedGridColumns Object) 178  
Add method (TListSelectedGridRows Object) 178  
Add property 179  
AddAfter method 181  
Adding items 32, 46, 179, 182, 183, 185, 247, 248, 290, 325, 326  
AddItem method 182, 215, 343, 345  
AddItem2 Method 183  
AddItem2Ex method 183  
AddRow method 185  
AfterEditing Event 186  
Align property 187  
Alignment property 226  
AllowResizing property 188  
Appearance property 189  
Appearance property (TListCheckbox Object) 189  
Associated Data 35  
AutoDragComplete event 190  
AutoDragMode property 192

AutoDragRequest event 191  
AutoExpand property 20, 192  
AutoFillColTitles property 193  
AutoFillRowTitles property 193  
Automating drag & drop 48  
Automating drag & drop 48  
AutoNavigate 439  
AutoNavigate object 317, 433  
AutoNewPage property 193  
AutoScrDuringDragDrop Property 47, 194  
AutoSizeColumn Method 194  
AutoSizeOptions Property 195  
AutoSizeRow Method 194  
**B**  
BackColor and SelBackColor Properties 197  
BackColor property 197  
BackColorBkg property 198  
Background 428, 432  
BackPicture property 198  
BackPictureAlignment property 198  
Backward Compatibility property 199  
BeforeDrag method 199  
*BeforeGridCellDeactivate* event 199  
*BeforeGridRowDeactivate* event 199  
BeforeItemDeactivate event 199  
BeginPage event 203  
Bookmarks 65, 225, 290, 294, 297, 306, 422, 423  
BorderColor property 203  
BorderStyle property 203  
BorderStyle Property (TListGrid Object) 204  
BottomIndex property 206  
BottomMargin property 322  
**C**  
Calendar Editing 59  
Caption property 206  
Categories 66  
CellDef property 206  
CellEdit Property 207  
Cells property 208  
CellValue Property (TListComboItem Object) 208  
Checkbox editing 209, 298

- Checkbox Editing 60
- Checkbox properties 189
- CheckBox property 209
- CheckboxValue property 209
- CheckedPicture property (TListCheckbox Object) 210
- CheckedValue property (TListCheckbox Object) 211
- Children Property 212
- ChildrenCount Property 212
- Clear method 217, 218
- Clear method (TListComboItems Object) 218
- Clear method (TListNodes Object) 217
- Clear method (TListSelectedGridCells Object) 218
- Clear method (TListSelectedGridColumns Object) 218
- Clear method (TListSelectedGridRows Object) 218
- Clear method (TVWSelectedNodes Object) 219
- ClearItem property 219
- Click event 199, 219
- Clipboard 220, 221, 421, 423
- Clipboard property 52, 220
- CoerceIndex property 220
- Col property 213
- ColDefs Property 214
- ColDelimiter property 182, 214, 215
- Collapse event 221
- Collapsing 431
- Colors 17, 34, 256, 297
- Cols property 214
- ColTitleCellDef property 215
- ColTitlesHeight property 215
- Column Width 75, 194, 195
- Columns 21, 37, 417
- ComboBox property 215
- ComboBoxEditing 62
- Compatibility 245, 343
- ConvertTabsToCols property 182, 215
- Coordinates 269, 422
- CopyBuffer method 221
- Copying Between TList Controls 76
- CopyItem property 221
- CopyItemSub property 222
- CopyOne property 223
- CopySelected property 223
- Count property 216
- Count property (TListSelectedGridCells Object) 216
- Count property (TListSelectedGridColumns Object) 216
- Count property (TListSelectedGridRows Object) 216
- Count property (TVWSelectedNodes Object) 217
- CurrentIndexMethod property 24, 182, 220, 223, 224, 424, 428
- CurrentItem property 224
- CurrentItemBM property 225
- CurrentParent property 224
- D**
- Databases - working with 43
- Date/Time Editing 59
- DateTime property 226
- DbClick event 229
- Default 418
- DefItemCellAlignment property 226
- DefItemCellBackColor property 197
- DefItemCellBorderColor property 203
- DefItemCellBorderStyle property 203
- DefItemCellDef property 229
- DefItemCellPictureAlignment property 227
- DefItemCellTextAlignment property 228
- DefMultiLine property 229
- Design-time Support 18, 88, 89, 90, 92, 93, 94, 95, 96, 99, 102, 105, 106, 107, 108, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119
- DisableNoScroll property 230
- Display 18, 31, 285, 287, 291, 292, 329, 330, 347, 348, 365, 366, 367, 374, 387, 403, 404, 406, 428, 429, 430, 431, 437, 438, 443
- DisplayPic Property (TListComboItem Object) 231

- DisplayValue Property  
(TListComboItem Object) 230
- Distribution Notes 12
- Double-Byte Characters 29
- Drag Drop 47, 48, 190, 191, 192, 234,  
235, 238, 249, 349, 350, 351, 352,  
353, 354, 355, 356, 357
- Drag method 232
- DragColumnsEnabled property 232
- DragDrop 233
- DragDrop event 233
- DragDropEx event 233
- DragHighlight property 234
- DragIcon property 234
- DragIconStyle property 235
- DragMode property 235
- DragOver event 233
- DragOverEx event 233
- DrawFocusRect property 236
- DropDownItemHeight property  
(TListComboBox) 236
- DropDownMaxHeight property  
(TListComboBox) 237
- DropDownWidth property  
(TListComboBox) 237
- DropTarget property 238
- E**
- Editable property 239
- EditableStartOptions Property 240
- EditAreaMaxHeight property  
(TListComboBox) 238
- EditAreaMaxWidth property  
(TListComboBox) 238
- EditAreaMinHeight property  
(TListComboBox) 238
- EditAreaMinWidth property  
(TListComboBox) 238
- EditInfo property 240
- EditInfoObject Property  
(TListEditingChangeInfo object 243
- Editing 54, 76, 148, 186, 207, 241, 242,  
275, 276, 277, 278, 300, 301, 380
- EditingKeyDown event 241
- EditingKeyPress event 241
- EditingKeyUp event 241
- EditingMode Property 242

- Enabled property 243
- EndPage event 244
- EnumIndex Property 244
- Environment Property 245
- Error messages 445
- Even and Odd Rows 43
- Events 132, 431
- Expand event 245
- Expand property 20, 245
- ExpandChildren property 246
- ExpandEx property 20, 247
- Expanding and Collapsing 20, 192, 245,  
246, 247, 248, 367, 369
- ExpandNewItem property 247
- ExpandToLevel property 248
- ExplorerCompatible property 248
- F**
- FastAddItem method 248
- FastAddItemEx method 248
- File I/O 53, 249, 325, 326, 384, 385,  
386, 423, 424
- File property 249
- Finding Item's Parent 27
- FindItem Method 250
- FindValue Method 250
- FireTListEvents property  
(TListTreeView) 251
- FirstItem property 251
- FirstSibling and LastSibling Properties  
251
- FixedSize property 252, 347
- Focus 236, 270
- FocusRectStyle property 252
- Font property 253
- Font... properties 253
- Font3D property 255
- FontName property 254
- Fonts 34, 253, 254, 255, 302
- FontShadowColor property 255
- FontShadowSelectedColor property 255
- FontSize property 254
- ForeColor property 256
- Format property 257
- Format property (TListDateTime  
Object) 261

FormatString property (TListDateTime Object) 263

Formatting 34

FoxPro support 266

FreeBuffer method 265

FullItemString and FullRowString Properties 265

FullPath property 266, 364

Functions 137

## **G**

GetArrayProperty

SetArrayProperty, GetArrayPropertyID Properties 266

GetArrayProperty, SetArrayProperty, GetArrayPropertyID Properties 266

GetData method 267

GetFormat method 268

GetItemByCellValue method (TListComboItems) 268

GetItemByXY function 269

GetItemRect method 270

GotFocus event 270

GradientColorFrom property 271

GradientColorTo property 271

GradientStyle property 271

GrayedPicture property (TListCheckbox Object) 210

GrayedValue property (TListCheckbox Object) 211

Grid 21, 37, 185, 188, 193, 198, 204, 207, 208, 214, 215, 229, 232, 265, 272, 273, 274, 278, 279, 280, 281, 282, 303, 304, 341, 364, 379, 382, 383, 396, 399, 405, 410, 429, 438

Grid Cells Editing support 207, 242, 275, 276, 277, 278

Grid property 272

GridCellActivate Event 272

GridCellAfterEditing Event 276

GridCellAfterEditing Event 276

GridCellClick event 273

GridCellDbClick event 274

GridCellDeactivate Event 272

GridCellDef property 274

GridCellEditingChange Event 277

GridCellEditingChange Event 277

GridCellEditingKeyDown

GridCellEditingKeyUp and GridCellEditingKeyPress Events 278

GridCellEditingKeyDown, GridCellEditingKeyUp and GridCellEditingKeyPress Events 278

GridCellRequestEditing Event 275

GridCellRequestEditing Event 275

GridColumnTitleClick Event 281

GridCornerTitleClick Event 281

GridLines3DLightColor and GridLines3DShadowColor Properties 280

GridLines3DStyle property 279

GridLinesColor property 278

GridLinesStyle property 279

GridRowActivate Event 281

GridRowDeactivate Event 281

GridRowTitleClick Event 281

## **H**

HasCell property 282

HasGrid property 283

HasSubItems property 283

Height property 283

HeightScale property (TListTextBox Object) 332

HelpContextID property 284

Hidden data 35

Hidden Items 20, 296, 329, 404

HitTest method 318

Hot Spots 31

How to 17, 20, 21, 24, 26, 27, 29, 32, 33, 34, 35, 36, 37, 43, 46, 47, 48, 49, 50, 51, 52, 53, 54, 65, 66, 67, 69, 70, 72, 73, 74, 75, 76

HScroll event 284

hWnd property 285

## **I**

Image property 285

ImageStretch property 287

Indent property 287, 402

Indentation property 288

Index property 289

Index Property(TListNode Object) 288

IndexByBM method 290

Indexes 24, 424, 428

In-place Editing 54, 148, 186, 207, 241, 242, 275, 276, 277, 278, 300, 301, 380

- Insert property 290
- InsertItem property 290
- IntelliMouse support 74, 342, 441
- Internet 22, 69
- InvBorderStyle property 291
- InvImage property 291
- InvStyle property 292
- IsClipboardAvailable Property 293
- IsItemVisible property 293
- IsPrinting method 294
- IsValidBM method 294
- IsValidBuffer method 294
- Item method 295
- Item method (TListSelectedGridCells Object) 295
- Item method (TListSelectedGridColumns Object) 295
- Item method (TListSelectedGridRows Object) 295
- Item property 308
- Item Property (TVWSelectedNodes Object) 296
- Item...Value properties 315
- ItemActivate Event 296
- ItemAlwaysHidden property 296
- ItemBackColor property 297
- ItemBM 65
- ItemCell property 298
- ItemCheckboxValue property 298
- ItemClick event 299
- ItemData replacement 35
- ItemDblClick event 299
- ItemDeactivate Event 296
- ItemEditingChange Event 301
- ItemEditingChange Event 301
- ItemEditText property 300
- ItemFont... properties 302
- ItemFontName property 302
- ItemFontSize property 302
- ItemForeColor property 297
- ItemGrid - Definition 21, 37
- ItemGrid property 303
- ItemHasGrid property 304
- ItemHasValue property 315
- ItemHeight property 304
- ItemImageDefHeight property 304
- ItemImageDefWidth property 304
- ItemIndex property 303
- ItemIndexToRow method 303
- ItemLastSubItemIndex **property** 26, 307
- ItemMark property 305
- ItemMultiLine property 305
- ItemNextSibling property 26, 307
- ItemParent property 26, 306
- ItemParentBM property 306
- ItemPMPicType property 307
- ItemPrevSibling property 307
- ItemQueryData event 308, 376
- Items property (TListComboBox Object) 309
- ItemSorted property 310
- ItemSortingKey property 310
- ItemSortingMethod property 310
- ItemSortingStyle property 310
- ItemTag property 313
- ItemToolTip property 314
- ItemType property 315
- ItemUrl property 317
- ItemValue property 35, 315
- ItemVirtualCount property 316
- ItemVirtualParent property 316
- ItemXXXValue replacement 35
- K**
- Keyboard Interface 22
- KeyboardActivation Property (TList object) 319
- KeyboardNavigateWhileEditing Property 321
- KeyDown event 320
- KeyPress event 320
- KeyUp event 320
- L**
- Languages 245
- LastItem property 251
- Left property 322
- LeftMargin property 322
- LevelDefs and Sorting 73
- LevelDefs property 322
- License Registration 8
- Licensing Restrictions 12
- List property 323

- ListCount property 323
- ListCountEx property 324
- ListIndex property 324
- LoadAndAdd property 325
- LoadAndInsert property 325
- LoadBuffer method 326
- LoadData method 326
- LoadData method (TListTreeView) 327
- Loading Trees 53, 249, 325, 326, 423
- LoadPicture method 327
- LostFocus event 328
- M**
- Manual Drag Drop 49
- MarginBottom Property 328
- MarginLeft Property 328
- MarginRight Property 328
- MarginTop Property 328
- Mark Array 66
- MarkClick event 328
- MarkDbClick event 328
- MarkedItemsAlwaysHidden property 329
- MarkHeight property 330
- MarkPicture property 66, 329
- Marks 66, 305, 328, 329, 330, 367, 404
- MarkTag property 66, 330
- MarkWidth property 330
- Max property (TListDateTime Object) 331
- Max property (TListSpin Object) 331
- MaxHeight property (TListTextBox Object) 332
- MaxLength property (TListTextBox Object) 330
- MaxWidth property (TListTextBox Object) 333
- Memory 265, 422
- Menu 69
- Methods 135
- Min property (TListDateTime Object) 331
- Min property (TListSpin Object) 331
- MinHeight property (TListTextBox Object) 332
- MinWidth property (TListTextBox Object) 333

- Modifications property 334
- MouseCol property 341
- MouseDown event 341
- MouseIcon property 342
- MouseMove event 341
- MousePointer property 342
- MouseRow property 341
- MouseUp event 341
- MouseWheel** Event 342
- Move event 269, 422
- Move method 343
- MoveItem method 339
- MoveTo method 340
- Moving items 288, 339, 363
- MSOutlineAdd property 182, 343
- MSTreeView and TListTreeView 83
- MultiLine property 305, 344, 366, 442, 443
- MultiSelect property 29, 344
- N**
- Name property 345
- Navigating 24
- Navigation 76
- New in Version 6! 73, 74, 148, 181, 204, 207, 242, 243, 265, 275, 276, 277, 278, 279, 296, 301, 334, 339, 342, 346, 348, 381, 401, 431, 434, 435, 441
- New in Version 7! 39, 51, 162, 163, 164, 172, 173, 174, 176, 178, 193, 194, 195, 216, 218, 232, 233, 252, 255, 257, 272, 273, 281, 295, 310, 328, 349, 350, 352, 354, 355, 356, 377, 382, 388, 390, 391, 392, 394, 395, 396, 399, 410, 420, 430, 431
- New in version 8! 8, 76, 166, 167, 168, 169, 188, 199, 314, 321, 347, 374, 388, 407
- NewIndex property 345
- Next and Prev 346
- Node Property 346
- NodeFromItem
  - NodeFromGridCell, NodeFromTListNode Methods (TListTreeView object) 346
- NodeFromItem, NodeFromGridCell, NodeFromTListNode Methods (TListTreeView object) 346
- Nodes (TListTreeView) 424



- NoIntegralHeight property 347
- NonScrollableColumns property 347
- NoPictureRoot property 348, 365
- O**
- Objects 121, 123, 137, 139, 140, 141, 142, 144, 145, 146, 147, 148, 149, 152, 153, 154, 156, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169
- Objects Reference 121
- OldSelStart And OldSelLength
  - Properties(TListEditingChangeInfo object 348
- OldValue Property
  - (TListEditingChangeInfo object 434
- OLE Drag Drop 50, 249, 349, 350, 351, 352, 353, 354, 355, 356
- OLECompletedDrag event 349
- OLEDrag method 350
- OLEDragDrop event 351
- OLEDragMode property 350
- OLEDragOver event 353
- OLEDropMode property 352
- OLEGiveFeedback event 354
- OLESetData event 356
- OLEStartDrag event 355
- OnDragDrop method 357
- OnDragOver method 357
- Options property (TListCheckbox Object) 357
- Options property (TListComboBox Object) 361
- Options property (TListDateTime Object) 358
- Options property (TListSpin Object) 359
- Options property (TListTextBox Object) 360
- P**
- Pages property 362
- Palette 29
- Parent 27
- Parent property 364
- Parent Property (TListNode Object) 363
- ParentItemIndex property 364
- PasteBuffer method 363
- PathSeparator property 364

- Performance 33, 36
- PicInMultiLine property 366
- Picture property 365
- Picture... properties 365, 366
- PictureAlignment property 227
- PictureClick event 199, 367
- PictureDblClick event 367
- PictureHeight property 368
- PictureMark property 367
- PictureMinus property 367
- PicturePalette property 366
- PicturePlus property 367
- Pictures 29
- PictureSelected property 365
- PictureType property 368
- PictureWidth property 368
- PlusMinus pictures 192, 307, 367
- PlusMinusClick event 369
- PlusMinusDblClick event 369
- PostScriptDC property 369
- PrepareForPrinting method 370
- PreparePage event 370
- PreviewMode property 370
- PreviewPageHeight property 371
- PreviewPageWidth property 371
- PrintBackground property 371
- PrinterObject property 372
- Printing 70, 156, 158, 159, 171, 193, 203, 244, 251, 294, 322, 362, 369, 370, 371, 372, 380, 405, 414, 443
- PrintingStop method 372
- PrintLevels property 371
- PrintOneStep method 372
- Properties 123
- Property Array Index 24
- R**
- RecalculateSize Method
  - (TListTooltipWindow Object) 374
- Redraw property 374
- Redraw property (TListTreeView) 375
- Refresh method 374, 375, 376
- RefreshItems method 376
- Remove method 377, 378
- Remove method (TListComboItems Object) 376

- Remove method
  - (TListSelectedGridCells Object) 377
- Remove method
  - (TListSelectedGridColumns Object) 377
- Remove method
  - (TListSelectedGridRows Object) 377
- RemoveItem method 379
- RemoveItem method (TListComboItems Object) 379
- RemoveRow method 379
- Removing items 32, 46
- Report 380
- Report property 380
- RequestEditing event 380
- Resize Columns 75, 194, 195
- Resize Rows 75, 194, 195
- Right Mouse Clicks 299
- RightMargin property 322
- Right-Mouse Menu 69
- Root Property 381
- Row property 213
- RowCellDef property 43, 382
- RowDefs Property (TListGrid object) 382
- RowHeight property 382
- Rows property 214
- RowTitleCellDef property 383
- RowTitlesWidth property 383
- RowToItemIndex method 383
- RTF 72
- RTFStyle property 384
- S**
- SafeAdd method (TListComboItems Object) 177
- Save property 384
- SaveBuffer method 385
- SaveData method 385
- SaveData method (TListTreeView) 386
- SaveOne property 386
- SaveSub property 386
- Saving Trees 53, 249, 384, 385, 386, 424
- Scrolling 76
- Scrollbars property 387
- ScrollHorz property 387
- ScrollHPosition property 388

- ScrollHRange property 388
- Scrolling 47, 284, 387, 388, 390, 427
- ScrollVModeKeyboard property 388
- ScrollVModeMouse property 388
- ScrollVModeMouse,
  - ScrollVModeKeyboard,
  - ScrollVStepMouse and
  - ScrollVStepKeyboard properties 388
- ScrollVPosition property 390
- ScrollVRange property 390
- ScrollVStepKeyboard property 388
- ScrollVStepMouse property 388
- Searching 51, 250, 421
- SelBackColor property 391
- SelBorderColor property (TListCellDef object) 392
- SelBorderStyle property (TListCellDef object) 391
- Selectable Property (TListCellDef object) 392
- Selected property 393
- Selected Property (TListGridCell TListRowDef and TListColDef objects) 394
- Selected Property (TListGridCell, TListRowDef and TListColDef objects) 394
- SelectedCells Property 394
- SelectedColumns Property 395
- SelectedItem Property (TListTreeView) 395
- SelectedNodes Property (TListTreeView) 395
- SelectedRows Property 395
- SelectEx property 396
- Selection 29, 344, 391, 393, 396, 400, 401
- Selection (TListTreeView) 165, 181, 217, 219, 296, 378, 395, 399
- SelectionMode property (TListGrid object) 396
- SelectionMode Property (TListTreeView) 399
- SelectionOptions Property (TListGrid object) 399
- SelForeColor property 391
- SelItemCount property 400
- SelItemIndex property 401

- SelStart And SelLength
  - Properties(TListEditingChangeInfo object) 401
- SetFocus property 402
- Shift property 402
- ShiftStep property 403
- ShowCaption property 404
- ShowChildren property 404
- ShowColTitles property 405
- ShowColumnTitles property 405
- ShowHiddenItems property 404
- ShowRowTitles property 405
- ShowTitles property 406
- ShowTooltip Event 407
- SmartDragDrop property 406
- Smooth Scrolling 76
- Sorted property 410
- Sorting 51, 310, 410
- Sorting property (TListComboItems Object) 408
- SortingKey property 410
- SortingKey property (TListComboItems Object) 408
- SortingMethod property 410
- SortingStyle property 410
- SortingStyle property (TListComboItems Object) 409
- Special Colors 17
- Special Pictures 29
- Speed 33
- Spin Control Editing 59
- Spin property 414
- Start method 414
- States property (TListCheckBox Object) 415
- Step property (TListSpin Object) 414
- Storing Trees 53, 249, 384, 385, 386, 424
- Style property (TListComboBox Object) 416
- Style property (TListEditInfo Object) 415
- T**
- TabIndex property 416
- Tabs 417
- TabStop property 417

- TabStopDistance property 417
- Tag property 418
- TDesigner 18, 88, 89, 90, 92, 93, 94, 95, 96, 99, 102, 105, 106, 107, 108, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119
- Technique 17, 20, 21, 22, 24, 27, 29, 32, 33, 34, 35, 36, 37, 43, 46, 47, 48, 49, 50, 51, 52, 53, 54, 65, 66, 67, 69, 70, 72, 73, 74, 75, 76
- Text property 418
- TextAlignment property 228
- TextBox Editing 58
- TextBox property 419
- TitleHeight property 419
- TitlePicture property 419
- Titles 405, 406, 419, 429
- TitlesResize Event 420
- TitleText property 419
- TitleVisible property 419
- TitleWidth property 419
- TList Grids 21, 37
- TList Indexes 24
- TList Object 123
- TList Property (TListTreeView) 420
- TListCellDef Object 137, 172, 197, 203, 253, 255, 256, 257, 328, 344, 365, 391, 392, 433
- TListCellDef property 226, 227, 228, 271, 368, 384, 418
- TListCheckBox Object 139, 189, 209, 210, 211, 298, 357, 415
- TListColDef object 140, 206, 214, 272, 289, 340, 394, 433, 438, 441
- TListColDefs object 216, 308
- TListColDefs object collection 141, 214
- TListComboBox Object 141, 142, 144, 177, 215, 218, 309, 361, 376, 379, 408, 409, 416
- TListComboItem Object 142, 208, 230, 231
- TListComboItems Object 142, 144, 177, 218, 309, 376, 379, 408, 409
- TListCopyBuffer function 221, 421
- TListDataObject object 145, 249, 267, 268

TListDataObjectFiles object 146, 249  
 TListDateTime Object 147, 226, 261, 263, 331, 358  
 TListEditInfo Object 148, 209, 215, 226, 414, 419  
 TListEditingChangeInfo Object 148  
 TListFindItem function 421  
 TListFindValue function 421  
 TListFreeBuffer function 265, 422  
 TListGetItemByXY function 422  
 TListGetItemRect function 422  
 TListGrid object 149, 173, 174, 176, 185, 188, 193, 198, 204, 207, 208, 213, 214, 215, 265, 274, 278, 279, 280, 282, 303, 341, 364, 379, 382, 383, 394, 395, 396, 399, 405, 410, 429, 438  
 TListGridCell object 152, 206, 213, 272, 394, 433  
 TListIndexByBM function 290, 422  
 TListIsClipboardAvailable function 423  
 TListIsValidBM function 294, 423  
 TListIsValidBuffer function 294, 423  
 TListLevelDef object 153, 206, 288, 365, 410  
 TListLevelDefs object 216, 308  
 TListLevelDefs object collection 154  
 TListLoadBuffer function 326, 423  
 TListNode Object 154, 180, 197, 206, 212, 244, 245, 251, 272, 285, 287, 288, 291, 315, 316, 346, 363, 393, 410, 418, 438  
 TListNodes and TListNode Objects 74  
 TListNodes Object 156, 180, 216, 217, 377  
 TListPage object 158  
 TListPages object collection 159  
 TListPasteBuffer function 363, 423  
 TListPoint Object 166  
 TListRectangle Object 166  
 TListReport object 156, 193, 294, 362, 369, 370, 371, 372, 380, 414  
 TListReportPage object 362  
 TListRowDef object 394  
 TListRowDef Object 164  
 TListRowDefs Object 164  
 TListSaveBuffer function 385, 424  
 TListSelectedGridCells Object 162  
 TListSelectedGridColumns Object 163  
 TListSelectedGridRows Object 163  
 TListShowTooltipArgs Object 167  
 TListSpin Object 159, 331, 359, 414  
 TListTextBox Object 160, 330, 332, 333, 360, 419  
 TListTooltip Object 167  
 TListTooltipStyle Object 168  
 TListTooltipWindow Object 169  
 TListTooltipWindow Object 374  
 TListTranslateIndex function 424, 428  
 TListTreeView 83, 84, 165, 251, 327, 343, 346, 375, 386, 395, 399, 420, 424, 432  
 TListTreeView and TList 84  
 TListValue object 161, 303, 433  
 TListValues object 216, 308  
 TLNode Property (TListTreeView) 424  
 Tool Tips 18, 425, 426  
 ToolTip property 314  
 Tooltips 407  
 ToolTips 18, 75, 166, 167, 168, 169, 314, 425, 426  
 ToolTipsBackColor property 425  
 ToolTipsDelay property 425  
 ToolTipsForeColor property 425  
 ToolTipsMode property 425  
 ToolTipsText property 426  
 ToolTipsViewStyle property 426  
 Top property 427  
 TopIndex property 427  
 TopMargin property 322  
 TranslateIndex method 428  
 Transparent Background 432  
 Transparent Bitmap 29  
 Transparent Color 17  
 TransparentBackground property 428  
 TransparentBitmap property 428  
 TransparentBitmap property 428  
 TransparentBitmapColor property 429  
 TransparentBitmapColor property 429  
*tree buffer* 46, 221, 222, 223, 265, 290, 294, 326, 363, 385, 421, 422, 423, 424  
 Tree Grid - Definition 21, 37

- Tree Lines 430, 431
- TreeGrid property 429
- TreeLinesColor property 429, 431
- TreeLinesHighlightColor property 431
- TreeLinesShadowColor property 431
- TreeLinesStyle property 430
- TreePictureTable 249
- TreeView Compatibility 83
- TriggerEvents property 431
- TVWSelectedNodes Object 165, 181, 217, 219, 296, 378
- U**
- UncheckedPicture property (TListCheckbox Object) 210
- UncheckedValue property (TListCheckbox Object) 211
- UpdateBackground method 432
- UpdateImages Method (TListTreeView) 432
- Upgrading TList projects 67
- URL 317, 433, 440
- Url property 433
- User defined data 35, 313, 315, 330
- V**
- Value property 433
- Value property 433
- Value Property (TListEditingChangeInfo object 434
- ValueName property 433
- ValueName property 433
- ValueType Property (TListEditingChangeInfo object 435
- Version 69
- Version property 436
- ViewStyle property 437
- ViewStyleEx property 437
- Virtual Items 36, 316, 376
- Visible property 438
- Visible property (TListNode Object) 438
- Visual Elements 18, 31
- VisualRoot property 72, 438
- VScroll event 284
- W**
- Web Support 22, 69
- WebAutoNavigate property 439
- WebGoBack method 440

- WebGoForward method 440
- WebNavigate method 440
- WebTargetFrame property 439
- WebURLBase property 439
- WheelScrolling Property 441
- Width property 441
- WidthOfText property 442
- WidthOfTextMin property 443
- WWW 317, 433, 439, 440
- X**
- XOffset property 443
- Z**
- Zoom property 443
- ZOrder method 444