Extended List View - List Overview

Overviews

[ndy]
Class List Overview

Classes

- EELVException
- TEnhListView
- TExtListColumns
    - EELVOldComCtl
- TEnhLVSaveSettings
- TExtListView
    - TCustomEnhListView
- TExtListColumn
- TExtLVSaveSettings

Extended List View - Class Hierarchy Overview

Overviews

[ndy]
Hierarchy Overview

Class Hierarchy

- EELVException
    - EELVOldComCtl
- TCustomEnhListView
    - TEnhListView
    - TExtListView
- TEnhLVSaveSettings
    - TExtLVSaveSettings
- TExtListColumn
- TExtListColumns

Extended List View - Unit Overview

Overviews

[ndy]
Unit Overview

Unit Overview

EnhListView
ExtListView

Overviews

Unit EnhListView Overview

Overviews

The EnhListView.pas provides the TEnhListView component, it's base class TCustomEnhListView, and various supporting classes and types.

Classes

- TCustomEnhListView
- TEnhListView
- TEnhLVSaveSettings

Unit ExtListView Overview

Overviews

The ExtListView.pas provides the TExtListView component and various supporting classes and types.

Classes

- EELVException
- TExtListColumn
- TExtListView
    - EELVOldComCtl
- TExtListColumns
- TExtLVSaveSettings

EELVException Class

Methods Properties Hierarchy Overviews

Ancestor
Exception

Unit
ExtListView

Description
EELVException is a base exception class for the TExtListView component.   Any exceptions that are specifically raised in the component will be of this class type, or a descendant of this class.

EELVException Hierarchy

Exception

    EELVException

EELVOldComCtl Class

Methods Properties Hierarchy Overviews

Ancestor
EELVException

Unit
ExtListView

Description
EELVOldComCtl is an exception thrown by the TExtListView component when the TExtListView.RequireComCtlUpdate property is set to true and the the user's COMCTL32.DLL is less than version 4.70.0.0.

EELVOldComCtl Hierarchy

- Exception
  - EELVException
    - EELVOldComCtl

TCustomEnhListView Class

Methods Properties Hierarchy Overviews

Ancestor
TCustomListView

Unit
EnhListView

Description
TCustomEnhListView is a base class for the TEnhListView and TExtListView components.   It is used to define all availble properties and methods, but leaves the properties declared in the protected section. This allows you to write your own descend components from it and leave hidden any properties that aren't needed or wanted.

TCustomEnhListView Methods

- BeginUpdate
- LoadSettings
    - DefaultSort
- Resort
    - Destroy
- StoreSettings
    - EndUpdate

TCustomEnhListView Properties

- ActualColumn

TCustomEnhListView Hierarchy

TCustomListView

|

TCustomEnhListView

BeginUpdate Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
procedure BeginUpdate; virtual;
```

Visibility
public

Description
The BeginUpdate method is very similar to the TListItems.BeginUpdate property; it inhibits screen updates until EndUpdate is called. Where it differs is that it will also inhibit sorting as well. This is very useful when adding a large number of items, since sorting would normally occur every time an item was added. Using this method, sorting will not occur until EndUpdate is called, and then only if it is needed, i.e. an item has been added or changed. For this reason, you should always call this method instead of Items.BeginUpdate.

TCustomEnhListView.BeginUpdate See Also

EndUpdate

DefaultSort Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
procedure DefaultSort(ColumnIndex: integer; Descending: boolean); virtual;
```

Visibility
public

Description
DefaultSort causes the list view to be sorted according to the column index and direction passed. It will take into account the TEnhListView.LastColumnClicked value if TEnhListView.AutoColumnSort is set to acsSortToggle and reverse the direction in the Ascending parameter if ColumnIndex matches LastColumnClicked. This allows it to act the same as if the user had clicked on the column.

TCustomEnhListView.DefaultSort See Also

TEnhListView.AutoColumnSort
TEnhListView.AutoResort
TEnhListView.AutoSortAscending
TEnhListView.AutoSortStyle
TEnhListView.CurrentSortAscending
TEnhListView.LastColumnClicked

Destroy Method
Overviews

Applies To
TCustomEnhListView

Declaration
```
destructor Destroy; override;
```

Visibility
public

Description
Destroy destroys an instance of TCustomEnhListView.   It frees all internal memory allocations and releases any resources back to the system.

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the object is not already freed and only then calls Destroy.

EndUpdate Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
procedure EndUpdate; virtual;
```

Visibility
public

Description
The EndUpdate method is very similar to the TListItems.EndUpdate property; it is used with BeginUpdate to inhibit screen updating.   Where it differs is that it will also inhibit sorting as well.   This is very useful when adding a large number of items, since sorting would normally occur every time an item was added. Using this method, sorting will not occur until EndUpdate is called, and then only if it is needed, i.e. an item has been added or changed.   For this reason, you should always call this method instead of Items.BeginUpdate.

TCustomEnhListView.EndUpdate See Also

BeginUpdate

LoadSettings Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
function LoadSettings: boolean; virtual;
```

Visibility
public

Description
The LoadSettings method is used to load the width of all columns in the list view according to the values in the TEnhListView.SaveSettings property.

This method is always called, even if the TEnhLVSaveSettings.AutoSave property is false.   The values are simply not loaded in this case.

This method is called automatically when the component's window is created, so you should normally not need to call this method in your code.

TCustomEnhListView.LoadSettings See Also

StoreSettings
TEnhLVSaveSettings

Resort Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
procedure Resort; virtual;
```

Visibility
public

Description
The Resort method causes the list view to sort the data according to last column clicked by the user (accessible through the TEnhListView.LastColumnClicked property) and the TEnhListView.AutoColumnSort mode.

If this method is called while inside a BeginUpdate / EndUpdate, no sorting occurs until the final EndUpdate is called.   Rather, an interal flag is set to indicate that sorting will need to be done.

If the TEnhListView.AutoResort property is set, you should never need to call this method.

TCustomEnhListView.Resort See Also

TEnhListView.AutoColumnSort
TEnhListView.AutoResort
TEnhListView.AutoSortAscending
TEnhListView.AutoSortStyle
TEnhListView.CurrentSortAscending
TEnhListView.LastColumnClicked

StoreSettings Method

Overviews See Also

Applies To
TCustomEnhListView

Declaration
```
function StoreSettings: boolean; virtual;
```

Visibility
public

Description
The StoreSettings method is used to save all the values that are available. For TEnhListView components, only column widths are available.

This method is always called, even if the TEnhLVSaveSettings.AutoSave property is false. The values are simply not saved in this case.

This method is called automatically when the component's window is destroyed, so you should normally not need to call this method in your code.

TCustomEnhListView.StoreSettings See Also

LoadSettings
TEnhLVSaveSettings

ActualColumn Property

Overviews See Also

Applies To
TCustomEnhListView

Declaration
`property ActualColumn [Index: integer]: TListColumn;`
Read only property

Visibility
public

Description
The ActualColumn property is used to account for re-ordered columns. TEnhListView does not support re-ordered columns, only TExtListView does. But, there are several pieces of code in TEnhListView that are called by TExtListView, and therefor it needs to be aware of this situation. So, for TEnhListView, this property always returns the same thing as the Columns property.

For TExtListView, this property translates a column index such that it returns the column that matches what was set at design-time. For example, if there were three columns in this order:

Column 1	Column 2	Column 3

and lvxDragDropHeader was set in the TExtListView.ExtendedStyles property and the user had re-ordered the columns like this:

Column 2	Column 3	Column 1

then ActualColumn[2] would return the first column. This is needed so that your code can always find columns as they were created at design-time instead of trying to figure out where they were after the user re-ordered them.

TCustomEnhListView.ActualColumn See Also

TExtListView.ExtendedStyles

TEnhListView Class

Methods Properties Hierarchy Overviews

Ancestor
TCustomEnhListView

Unit
EnhListView

Description
TEnhListView is a list view control that provides enhanced functionality such as:

* Automatic sorting of text, numeric, and date columns
* Ability to automatically save and restore user column width settings
* Owner drawing events

This functionality does NOT require the updated COMCTL32.DLL from Microsoft.

TEnhListView Properties

- AutoColumnSort
- OnDrawSubItem
  - AutoResort
- OnEditCanceled
  - AutoSortAscending
- OnMeasureItem
  - AutoSortStyle
- OnSortBegin
  - CurrentSortAscending
- OnSortFinished
  - HeaderHandle
- OnSortItems
  - LastColumnClicked
- ReverseSortArrows
- OnAfterDefaultDrawItem
- SaveSettings
  - OnDrawHeader
- ShowSortArrows
  - OnDrawItem
- Style

TEnhListView Hierarchy

- TCustomListView
  - TCustomEnhListView
    - TEnhListView

AutoColumnSort Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property AutoColumnSort;
```

Visibility
published

Description
AutoColumnSort indicates how the list view should be sorted, if at all. It can be one of the following values:

· acsNoSort	No sorting should occur.
· acsSort	Sort in the direction indicated by the AutoSortAscending property.
· acsSortToggle	Initially sort in the direction indicated by the AutoSortAscending property, but reverse the direction any time the same column is resorted, i.e. when the user clicks the column header of the currently sorted column.

TEnhListView.AutoColumnSort See Also

AutoResort
AutoSortAscending
AutoSortStyle
CurrentSortAscending
OnSortBegin
OnSortFinished
OnSortItems
ShowSortArrows
TCustomEnhListView.DefaultSort
TCustomEnhListView.Resort

AutoResort Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property AutoResort;
```

Visibility
published

Description
AutoResort determines if the list view should automatically resort items when it determines that an item has been added or edited.

Note: If a large number of items will be changed when AutoResort is on, the changes should be made inside calls to TCustomEnhListView.BeginUpdate and TCustomEnhListView.EndUpdate. This will inhibit the resorting until after all changes are made instead of after each change. Also note that the Items.BeginUpdate and Items.EndUpdate methods do not do this.

TEnhListView.AutoResort See Also

AutoColumnSort
AutoSortAscending
AutoSortStyle
CurrentSortAscending
OnSortBegin
OnSortFinished
OnSortItems
ShowSortArrows
TCustomEnhListView.BeginUpdate
TCustomEnhListView.EndUpdate

AutoSortAscending Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property AutoSortAscending;
```

Visibility
published

Description
AutoSortAscending determines if sorting should be in ascending or descending order by default.

TEnhListView.AutoSortAscending See Also

AutoColumnSort
AutoResort
AutoSortStyle
CurrentSortAscending
OnSortBegin
OnSortFinished
OnSortItems
ShowSortArrows

AutoSortStyle Property

Overviews See Also

Applies To

TEnhListView

Declaration

```
property AutoSortStyle;
```

Visibility

published

Description

AutoSortStyle defines the method of sorting to be used. It can be one of the following values:

·	assSmart	"Smart" sorting will attempt to determine if the data is numeric or date and/or time and sort accordingly. If it determines that it is niether, it will sort it as a string.
·	assDefault	Normally list view style sorting; everything is sorted as a string.

TEnhListView.AutoSortStyle See Also

AutoColumnSort
AutoResort
AutoSortAscending
CurrentSortAscending
OnSortBegin
OnSortFinished
OnSortItems
ShowSortArrows

CurrentSortAscending Property

Overviews See Also

Applies To

TEnhListView

Declaration

```
property CurrentSortAscending;
```

Visibility

public

Description

CurrentSortAscending is a run-time only property that can be used to check the direction of the currently sorted column.

TEnhListView.CurrentSortAscending See Also

AutoColumnSort
AutoResort
AutoSortAscending
AutoSortStyle
OnSortBegin
OnSortFinished
OnSortItems
ShowSortArrows

HeaderHandle Property

Overviews

Applies To
TEnhListView

Declaration

```
property HeaderHandle;
```

Visibility
public

Description
HeaderHandle is the window handle of the list view column header window.   This value is mostly for internal use to enable things like owner drawn header controls, but is available publicly.

LastColumnClicked Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property LastColumnClicked;
```

Visibility
public

Description
LastColumnClicked property indicates the zero-based index of the currently sorted column. This is normally maintained internally and updated when the user clicks a column to sort it, or when the TCustomEnhListView.DefaultSort method is called. Assigning a value to the property will affect future sorting, but will not cause a resort automatically. It will change the sort indicator (ShowSortArrows) immediately, however.

TEnhListView.LastColumnClicked See Also

AutoColumnSort
CurrentSortAscending
TCustomEnhListView.DefaultSort
TCustomEnhListView.Resort

OnAfterDefaultDrawItem Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property OnAfterDefaultDrawItem;
```

Visibility
published

Description
The OnAfterDefaultDrawItem event fires after an item has been handled by default drawing in owner draw mode (Style = lvsOwnerDrawFixed). This is useful when you want to add something to what is normally drawn.

Control is the list view that fired the event.

ACanvas is the canvas that should be drawn on.

Index is the zero-based index of the item being drawn.

ARect describes the rectangle that should be drawn in.

State describes the state of the selected item. It can have any of the following values:

· odSelected	The item is selected.
· odDisabled	The entire list view is disabled.
· odFocused	The item currently has focus.

Note: Only vsReport ViewStyle supports owner drawing, so this event will not fire when the list view is in other viewing modes.

TEnhListView.OnAfterDefaultDrawItem See Also

OnDrawItem
OnDrawSubItem
OnMeasureItem
Style

OnDrawHeader Property

Overviews Example

Applies To
TEnhListView

Declaration
```
property OnDrawHeader;
```

Visibility
published

Description
The OnDrawHeader event fires when a column header needs to be drawn. This only applies to list views that are in the vsReport ViewStyle mode. If no handler is supplied for this event, default drawing will occur.

Control is the list view that needs the header item drawn.

ACanvas is the canvas that should be drawn on. Changes made to this canvas will be used for default drawing, if it is requested. This allows for simple elements such as color to be changed without having to actually do the drawing.

Index is the zero-based index of the header item being drawn.

ARect describes the rectangle that should be drawn in.

Selected indicates whether the header item is selected or not. Selected means that the header is depressed, i.e. the user has clicked on it and it is down.

DefaultDrawing is a variable parameter that can be set to true to indicate that the list view should draw this header item for you. This is useful when you are only interested in simply changing something with the canvas, such as the text color.

Note: The list view does **not** need to be owner drawn (Style = lvsOwnerDrawFixed) for this event to work.

TEnhListView.OnDrawHeader Example

TEnhListView.OnDrawHeader

This OnDrawHeader handler draws the header text in red if it is being pressed, and the currently sorted column header in italics.

```
procedure TForm1.AnExtListViewDrawHeader(Control: TWinControl;
  var ACanvas: TCanvas; Index: Integer; var ARect: TRect;
  Selected: Boolean; var DefaultDrawing: Boolean);
begin
  // Current sort column in italics
  if AnExtListView.LastColumnClicked = Index then
    ACanvas.Font.Style := ACanvas.Font.Style + [fsItalic];

  // Pressed headers in red text
  if Selected then
    ACanvas.Font.Color := clRed;

  // Tell the listview to draw the text for us.
  DefaultDrawing := TRUE;
end;
```

OnDrawItem Property

Overviews Example See Also

Applies To
TEnhListView

Declaration
```
property OnDrawItem;
```

Visibility
published

Description
The OnDrawItem event fires when an item needs to be drawn in owner draw mode (Style = lvsOwnerDrawFixed). If no handler is supplied for this event, default drawing will occur.

Control is the list view that needs the item drawn.

ACanvas is the canvas that should be drawn on. Changes made to this canvas will be used for default drawing, if it is requested. This allows for simple elements such as color to be changed without having to actually do the drawing.

Index is the zero-based index of the item being drawn.

ARect describes the rectangle that should be drawn in.

State describes the state of the selected item. It can have any of the following values:

·	odSelected	The item is selected.
·	odDisabled	The entire list view is disabled.
·	odFocused	The item currently has focus.

DefaultDrawing is a variable parameter that can be set to true to indicate that the list view should draw this item for you. This is useful when you are only interested in simply changing something with the canvas, such as the text color.

Note: Only vsReport ViewStyle supports owner drawing, so this event will not fire when the list view is in other viewing modes.

TEnhListView.OnDrawItem Example

TEnhListView.OnDrawItem

This OnDrawItem handler draws the even numbered rows with red text and the odd ones in purple, and selected rows in italics.

```
procedure TForm1.AnExtListViewDrawItem(Control: TWinControl;
  var ACanvas: TCanvas; Index: Integer; ARect: TRect;
  State: TOwnerDrawState; var DefaultDrawing, FullRowSelect: Boolean);
begin
  // Change selected font style
  if (odSelected in State) then
    ACanvas.Font.Style := ACanvas.Font.Style + [fsItalic]
  else
    ACanvas.Font.Style := ACanvas.Font.Style - [fsItalic];

  // Even in red, odd in purple
  if Odd(Index) then
    ACanvas.Font.Color := clPurple
  else
    ACanvas.Font.Color := clRed;

  // Tell the listview to draw the text for us.
  DefaultDrawing := TRUE;
end;
```

TEnhListView.OnDrawItem See Also

OnAfterDefaultDrawItem
OnDrawSubItem
OnMeasureItem
Style

OnDrawSubItem Property

Overviews Example See Also

Applies To
TEnhListView

Declaration
```
property OnDrawSubItem;
```

Visibility
published

Description
The OnDrawSubItem event fires when a subitem needs to be drawn in owner draw mode (Style = lvsOwnerDrawFixed). This event will not fire if there is an event handler for OnDrawItem and it does not specify default drawing should occur. If no handler is supplied for this event, default drawing will occur.

Control is the list view that needs the item drawn.

ACanvas is the canvas that should be drawn on. Changes made to this canvas will be used for default drawing, if it is requested. This allows for simple elements such as color to be changed without having to actually do the drawing.

Index is the zero-based index of the item being drawn.

SubItem is the zero-based index of the subitem being drawn. If the value is -1, the Caption text is being drawn instead of a subitem.

ARect describes the rectangle that should be drawn in.

State describes the state of the selected item. It can have any of the following values:

· odSelected	The item is selected.
· odDisabled	The entire list view is disabled.
· odFocused	The item currently has focus.

DefaultDrawing is a variable parameter that can be set to true to indicate that the list view should draw this item for you. This is useful when you are only interested in drawing certain subitems, or simply changing something with the canvas such as the color.

Note: Only vsReport ViewStyle supports owner drawing, so this event will not fire when the list view is in other viewing modes.

TEnhListView.OnDrawSubItem Example

TEnhListView.OnDrawSubItem

This OnDrawSubItem handler draws the even numbered columns in red and the odd ones in blue, and draws the first column (caption) in italics.

```
procedure TForm1.AnExtListViewDrawSubItem(Control: TWinControl;
  var ACanvas: TCanvas; Index, SubItem: Integer; ARect: TRect;
  State: TOwnerDrawState; var DefaultDrawing: Boolean);
begin
  if Odd(SubItem) then
    ACanvas.Font.Color := clBlue
  else
    ACanvas.Font.Color := clRed;

  // Italics for the first column
  if SubItem = -1 then
    ACanvas.Font.Style := ACanvas.Font.Style + [fsItalic]
  else
    ACanvas.Font.Style := ACanvas.Font.Style - [fsItalic];

  // Tell the listview to draw the text for us.
  DefaultDrawing := TRUE;
end;
```

TEnhListView.OnDrawSubItem See Also

OnAfterDefaultDrawItem
OnDrawItem
OnMeasureItem
Style

OnEditCanceled Property
Overviews

Applies To
TEnhListView

Declaration
```
property OnEditCanceled;
```

Visibility
published

Description
The OnEditCanceled event is fired when caption text of a list view item was being edited, but the changes were not accepted, i.e. the user pressed escape.

A normal TListView will only notify you when an item begins editing (OnEditing) and after it has been changed through editing (OnEdited). However, it will not fire any event to notify you if editing stops without the changes being accepted. This event corrects that shortcoming.

OnMeasureItem Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property OnMeasureItem;
```

Visibility
published

Description
The OnMeasureItem event fires the first time an owner draw mode (Style = lvsOwnerDrawFixed) list view is painted. It is used to determine the height of items in vsReport ViewStyle. If no handler is supplied for this event, the default height is used.

Control is the list view that needs to be measured.

AHeight is a variable parameter that is assigned the height of each item in the list view.

Note: Only vsReport ViewStyle supports owner drawing, so this event will not fire when the list view is in other viewing modes.

TEnhListView.OnMeasureItem See Also

OnAfterDefaultDrawItem
OnDrawItem
OnDrawSubItem
Style

OnSortBegin Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property OnSortBegin;
```

Visibility
published

Description
The OnSortBegin event fires just before a list view's items are to be sorted. This event will fire for list views that are automatcially sorted or sorted with the OnSortItems event.

Sender is the list view that has been sorted.

SortColumn is the zero-based index of the subitem that was sorted on. This value will be -1 if the Caption column is being sorted. This allows the value to be used as the SubItems array property.

Ascending indicates the direction of the sort.

TEnhListView.OnSortBegin See Also

AutoColumnSort
OnSortFinished
OnSortItems

OnSortFinished Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property OnSortFinished;
```

Visibility
published

Description
The OnSortFinished event fires after a list view's items have been sorted.   This event will fire for list views that are automatcially sorted or sorted with the OnSortItems event.

Sender is the list view that has been sorted.

SortColumn is the zero-based index of the subitem that was sorted on.   This value will be -1 if the Caption column is being sorted.   This allows the value to be used as the SubItems array property.

Ascending indicates the direction of the sort.

TEnhListView.OnSortFinished See Also

AutoColumnSort
OnSortBegin
OnSortItems

OnSortItems Property

Overviews Example See Also

Applies To
TEnhListView

Declaration
```
property OnSortItems;
```

Visibility
published

Description
The OnSortItems event fires whenever the list view needs to be resort. It replaces the OnCompare event of a normal TListView because that event does not include information about which column is being sorted. This event allows for more customized sorting when assDefault AutoSortStyle isn't enough.

Sender is the list view that needs to be sorted.

Item1 and Item2 are the TListItems that need to be compared.

SortColumn is the zero-based index of the subitem being sorted. This value will be -1 if the Caption column is being sorted. This allows the value to be used as the SubItems array property index for retrieve values for comparison.

CompResult is a variable parameter that is assigned the result of the comparison. If Item1 is less than Item2, assign a value of -1; if Item1 is greater than Item2, assign 1; if equal, assign 0. The list view will handle the direction of the sort automatically, you do not need to account for ascending or descending order.

TEnhListView.OnSortItems Example

TEnhListView.OnSortItems

This OnSortItems event handler knows how to sort numeric values that have a trailing string.   For example, "42 KB" is greater than "9 KB", which would not be true if sorted as strings.

```
procedure TForm1.AnExtListViewSortItems(Sender: TObject; Item1,
  Item2: TListItem; SortColumn: Integer; var CompResult: Integer);
var
  S1,
  S2: string;
  Value1,
  Value2: integer;
  SpacePos: integer;
begin
  if SortColumn < 0 then
  begin
    S1 := Item1.Caption
    S2 := Item2.Caption
  end else begin
    S1 := Item1.SubItems[SortColumn];
    S2 := Item2.SubItems[SortColumn];
  end;

  SpacePos := Pos(' ', S1);
  Value1 := StrToIntDef(Copy(S1, 1, SpacePos-1), 0);
  SpacePos := Pos(' ', S2);
  Value2 := StrToIntDef(Copy(S2, 1, SpacePos-1), 0);

  if Value1 > Value2 then
    CompResult := 1
  else if Value1 < Value2 then
    CompResult := -1
  else
    CompResult := 0;
end;
```

TEnhListView.OnSortItems See Also

AutoColumnSort
OnSortBegin
OnSortFinished

ReverseSortArrows Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property ReverseSortArrows;
```

Visibility
published

Description
ReverseSortArrows is used to control whether the sort arrows should point up or down for ascending sorts, and vice versa for descending sorts.   A value of false means that the arrow will point up for ascending, and true will point down.

TEnhListView.ReverseSortArrows See Also

ShowSortArrows

SaveSettings Property
Overviews

Applies To
TEnhListView

Declaration
```
property SaveSettings;
```

Visibility
published

Description
SaveSettings is used to describe what user settings should be stored and retrieved automatcially between sessions.   See the TEnhLVSaveSettings class for a complete description.

ShowSortArrows Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property ShowSortArrows;
```

Visibility
published

Description
ShowSortArrows indicates whether the column headers should show a triangular arrow in the currently sorted column that indicates the direction of the sort.

TEnhListView.ShowSortArrows See Also

AutoColumnSort
ReverseSortArrows

Style Property

Overviews See Also

Applies To
TEnhListView

Declaration
```
property Style;
```

Visibility
published

Description
The Style property indicates whether the list view is owner drawn or not. The possible values are:

· lvsStandard	A non-owner drawn list view.
· lvsOwnerDrawFixed	An owner drawn list view in which all items are the same height. Variable height owner drawing is not supported by the list view control, so there is no lvsOwnerDrawVariable value.

For owner drawn list views that do not have an event handler for the OnDrawItem or OnDrawSubItem event, the list view draw the items for you. This is effectively the same as lvsStandard style. The usefulness of this default drawing becomes apparent when you want only to make simple changes such as drawing certain lines in a different color. The drawing events allow you to make changes to the canvas and then specifiy that default drawing should occur. See those help topics and their examples for further information.

Note: Only vsReport ViewStyle supports owner drawing, so this event will not fire when the list view is in other viewing modes.

TEnhListView.Style See Also

OnAfterDefaultDrawItem
OnDrawSubItem
OnMeasureItem
Style

TEnhLVSaveSettings Class

Methods Properties Hierarchy Overviews

Ancestor
TPersistent

Unit
EnhListView

Description
The TEnhLVSaveSettings class is used to describe what elements of a TEnhListView to automatically save and restore, if any, and where to save those values in the registry.

TEnhLVSaveSettings Methods

- Create
- StoreColumnSizes
- ReadColumnSizes

TEnhLVSaveSettings Properties

- AutoSave
- SaveColumnSizes
  - RegistryKey

TEnhLVSaveSettings Hierarchy

TPersistent
  |
  TEnhLVSaveSettings

Create Method

Overviews

Applies To
TEnhLVSaveSettings

Declaration
```
constructor Create; virtual;
```

Visibility
public

Description
The Create constructor creates a new instance of the class and initializes internal variables.   It does nothing out of the ordinary that you should be concerned with.

ReadColumnSizes Method
Overviews

Applies To
TEnhLVSaveSettings

Declaration
```
procedure ReadColumnSizes(ColCount: integer; var IntArray: array of integer);
```

Visibility
public

Description
ReadColumnSizes is the method used to actually read the column width values from the registry. ColCount is the number of elements that the IntArray parameter can hold, and IntArray will be populated with the column widths.   The values are read from the registry key indicated by the RegistryKey property.

StoreColumnSizes Method
Overviews

Applies To
TEnhLVSaveSettings

Declaration
```
procedure StoreColumnSizes(ColCount: integer; const IntArray: array of
integer);
```

Visibility
public

Description
StoreColumnSizes is a method used to actually write the column width values to the registry.   ColCount is the number of elements being passed in the IntArray parameter, which holds the actual column widths. The values are stored in the registry key indicated by the RegistryKey property.

AutoSave Property
Overviews

Applies To
TEnhLVSaveSettings

Declaration
```
property AutoSave: boolean;
```

Visibility
published

Description
AutoSave indicates whether the values should be automatically saved and restored between sessions.

RegistryKey Property

Overviews

Applies To

TEnhLVSaveSettings

Declaration

```
property RegistryKey: string;
```

Visibility

published

Description

RegistryKey indicates the registry key under HKEY_CURRENT_USER to use for storing and reading saved values.   Each element stored will create a subkey under the specified key for its data.

SaveColumnSizes Property

Overviews See Also

Applies To
TEnhLVSaveSettings

Declaration
```
property SaveColumnSizes: boolean;
```

Visibility
published

Description
SaveColumnSizes indicates whether the column widths of the list view should be saved and restored between sessions.

TEnhLVSaveSettings.SaveColumnSizes See Also

AutoSave

TExtListColumn Class

Methods Properties Hierarchy Overviews

Ancestor
TCollectionItem

Unit
ExtListView

Description
TExtListColumn is a class used to describe the extended formatting attributes of a list view column made available by the COMCTL32.DLL v4.7x update. Instances of this class are accessed through the TExtListView.ColumnsFormat property.

TExtListColumn Methods

- Assign
- Destroy
- Create

TExtListColumn Properties

ImageAlignment

ImageIndex

TExtListColumn Hierarchy

TCollectionItem

|

TExtListColumn

Assign Method
Overviews

Applies To
TExtListColumn

Declaration
```
procedure Assign(Source: TPersistent); override;
```

Visibility
public

Description
Assign copies the values of the TExtListColumn instance passed in the Source parameter to this instance.

Create Method
Overviews

Applies To
TExtListColumn

Declaration
```
constructor Create(Collection: TCollection); override;
```

Visibility
public

Description
The Create constructor creates a new instance of the class and initializes internal variables.   It does nothing out of the ordinary that you should be concerned with.

Destroy Method
Overviews

Applies To
TExtListColumn

Declaration
```
destructor Destroy; override;
```

Visibility
public

Description
Destroy destroys an instance of TExtListColumn.   It frees all internal memory allocations and releases any resources back to the system.

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the object is not already freed and only then calls Destroy.

TExtListView maintains a list of TExtListColumns, and takes care of their allocation and deallocation as needed.   You should normally never need to create or destroy an instance yourself unless you are dynamically creating and destroying list columns at run-time.

ImageAlignment Property

Overviews See Also

Applies To
TExtListColumn

Declaration
```
property ImageAlignment: TColumnImageAlign;
```

Visibility
published

Description
ImageAlignment indicates which side of the column header text to draw the image on.

· ciaLeftOfText:	Draw image to the left of the text.
· ciaRightOfText:	Draw image to the left of the text.

TExtListColumn.ImageAlignment See Also

ImageIndex

ImageIndex Property

Overviews See Also

Applies To
TExtListColumn

Declaration
```
property ImageIndex: integer;
```

Visibility
published

Description
ImageIndex indicates the image number to use for this column header in the image list identified by the TExtListView.SmallImages property.

I understand that some of you don't like having to use the SmallImages property of the list view for the column header images, but this is how Microsoft wrote it. The only other option you have is to use owner drawn headers (available in TEnhListView) and draw the image youself from whatever source you like.

TExtListColumn.ImageIndex See Also

ImageAlignment

TExtListColumns Class

Methods Properties Hierarchy Overviews

Ancestor
TCollection

Unit
ExtListView

Description
TExtListColumns contains a collection of TExtListColumn items that describes the extended formatting to be applied to the list view columns. The TExtListView.ColumnsFormat property is used to access the instance of this list.

TExtListColumns Methods

- Add
- Create
    - Assign
- Refresh

TExtListColumns Properties

- Items
- Owner

TExtListColumns Hierarchy

TCollection

|

TExtListColumns

Add Method

Overviews

Applies To

TExtListColumns

Declaration

```
function Add: TExtListColumn;
```

Visibility

public

Description

Adds a new instance of TExtListColumn to this collection, and returns the instance in the function result.

Assign Method
Overviews

Applies To
TExtListColumns

Declaration
```
procedure Assign(Source: TPersistent); override;
```

Visibility
public

Description
Assign copies the values of the TExtListColumns instance passed in the Source parameter to this instance.

Create Method

Overviews

Applies To

TExtListColumns

Declaration

```
constructor Create(AOwner: TExtListView);
```

Visibility

public

Description

The Create constructor creates a new instance of the class, sets the owning TExtListView, and initializes internal variables.   It does nothing out of the ordinary that you should be concerned with.

Refresh Method

Overviews See Also

Applies To
TExtListColumns

Declaration
```
procedure Refresh;
```

Visibility
public

Description
Refresh is called to reset all extended column formatting information.

Changing formatting information (such as the TExtListColumn.ImageIndex or TExtListColumn.ImageAlignment properties) at run-time does not automatically update the display. I could catch this situation if Borland would have been a little more liberal with their virtual/dynamic declarations, but the one I need isn't declared that way, so I can't override it. Instead, you will have to be aware of the situation and manually refresh the display this way.

TExtListColumns.Refresh See Also

TExtListView.UpdateColumnImage

Items Property

Overviews

Applies To

TExtListColumns

Declaration

```
property Items [Index: Integer]: TExtListColumn;
```

Visibility

public

Description

Items provides access to the individual TExtListColumn formatting items.

Owner Property

Overviews

Applies To

TExtListColumns

Declaration

```
property Owner: TExtListView;
```

Read only property

Visibility

public

Description

Owner indicates the TExtListView that is associated with this set of formatting data.   It is used so that it can update the list view when changes are made.

This value is set when the collection is created, and normally never needs to be changed.

TExtListView Class

Methods Properties Hierarchy Overviews

Ancestor
TCustomEnhListView

Unit
ExtListView

Description
TExtListView is a list view control that provides all functionality of TEnhListView, plus provides access to the new features of the list view control found with MS Internet Explorer 4.0, et. al. This functionality **DOES** require the updated COMCTL32.DLL from Microsoft. The COMCTL32.DLL file must be version 4.72.xxx or later for all functionality in this component to work. Earlier versions *may* work, but are not supported (i.e. don't email me if you aren't using the latest version of COMCTL32.DLL).

Using this component on a system with earlier versions should have no ill effects, the new functionality simply won't be there. The one exception is VirtualMode which is not useable with old versions at all.

TExtListView Methods

- ApproximateViewRect
- SetColumnOrder
    - CheckComCtlVersion
- SetIconSpacing
    - ELV_GetNextItem
- SetItemCountEx
    - GetColumnOrder
- StoreSettings
    - GetSubItemAt
- UpdateColumnImage
    - LoadSettings
- UpdateColumnsImages

TExtListView Properties

- BackgroundImage
- OnVMFindItem
- ColumnsFormat
- OnVMGetItemInfo
- ExtendedStyles
- OnVMStateChanged
- HotCursor
- RequireComCtlUpdate
- HotItem
- SaveSettings
- HoverTime
- SmallImages
- IsChecked
- SubItem_BoundsRect
- ItemIndent
- SubItem_IconRect
- Items
- SubItem_ImageIndex
- OnHotTrack
- SubItem_LabelRect
- OnInfoTip
- SubItem_SelectBoundsRect
- OnItemActivate
- VirtualMode
- OnMarqueeBegin
- WorkArea
- OnVMCacheHint

TExtListView Hierarchy

- TCustomListView
  - TCustomEnhListView
    - TExtListView

ApproximateViewRect Method
Overviews

Applies To
TExtListView

Declaration
```
function ApproximateViewRect(Count: integer; const Proposed: TPoint): TPoint;
```

Visibility
public

Description
ApproximateViewRect calculates the approximate width and height required to display a given number of items.

Count contains the number of items to be displayed.   If this value is set to -1, the total number of items currently in the control is used.

Proposed should contain the proposed width and height of the control. Either value can be set to -1 to allow the function to use the current width or height value.

The approximate width and height needed to display the items is returned by the function.

Setting the size of the list view control based on the dimensions provided by this message can optimize redraw and reduce flicker.

CheckComCtlVersion Method

Overviews Example

Applies To
TExtListView

Declaration
```
function CheckComCtlVersion(MajorHi: word; MajorLo: word; MinorHi: word;
MinorLo: word): boolean;
```

Visibility
public

Description
The CheckComCtlVersion function reads the version information contained in the COMCTL32.DLL and compares the file version information to the values passed in MajorHi, MajorLo, MinorHi and MinorLo. If the version information value is equal to or higher than the values passed, the function returns true; otherwise, false is returned.

TExtListView.CheckComCtlVersion Example

TExtListView.CheckComCtlVersion

Check to see if version 4.70.0.0 or greater is installed.

```
  if AnExtListView.CheckComCtlVersion(4, 70, 0, 0) then
    { installed! };
  else
    MessageDlg('This application requires version 4.70.0.0 or greater ' +
       'of COMCTL32.DLL.', mtError, [mbOK], 0);
```

ELV_GetNextItem Method

Overviews Example See Also

Applies To
TExtListView

Declaration
```
function ELV_GetNextItem(StartItem: integer; Direction: TSearchDirection;
States: TItemStates): integer;
```

Visibility
public

Description
ELV_GetNextItem is similar to TListView's GetNextItem function except that it returns an item's index instead of a TListItem reference. This is included mainly for use with VirtualMode because it has no TListItems, i.e. GetNextItem always returns **NIL**.

By using ELV_GetNextItem, you can find list item indices that meet various conditions such as being selected, focused, etc. This is the only way to find things such as all selected items in a virtual mode list view.

StartItem identifies the index to begin the search from. The search does not include this item. Specifying -1 indicates the search should begin with the first item.

The Direction parameter can be one of the following values:

· sdLeft	GetNextItem looks to the left of the specified item.
· sdRight	GetNextItem looks to the right of the specified item.
· sdAbove	GetNextItem looks above the specified item.
· sdBelow	GetNextItem looks below the specified item.
· sdAll	GetNextItem looks in the order that items appear in the Items property.

The States parameter can take the following values:

· isNone	Only items in the default state are returned
· isCut	Only items marked for a cut and paste operation are returned.
· isDropHilited	Only items highlighted as a drag-and-drop target are returned
· isFocused	The value of the ItemFocused property is returned.
· isSelected	Only selected items are returned.

ELV_GetNextItem returns the next item index that matches the search, or -1 if no items match the given criteria.

TExtListView.ELV_GetNextItem Example

TExtListView.ELV_GetNextItem

Find all selected items in a virtual mode list view

```
var
  i: integer;
  s: string;
begin
  s := 'Selected indices: ';
  { Look for next item starting at the top (-1) }
  i := AnExtListView.ELV_GetNextItem(-1, sdAll, [isSelected]);
  while i > -1 do
  begin
    s := s + IntToStr(i) + ' ';
    { Look for next item starting at last index found }
    i := AnExtListView.ELV_GetNextItem(i, sdAll, [isSelected]);
  end;
  ShowMessage(s);
end;
```

TExtListView.ELV_GetNextItem See Also

VirtualMode

GetColumnOrder Method

Overviews Example See Also

Applies To
TExtListView

Declaration
```
function GetColumnOrder(Count: integer; var IntArray: array of integer):
boolean;
```

Visibility
public

Description
GetColumnOrder retrieves the current left-to-right order of columns in a list view control. The function returns a boolean value to indicate success or failure.

Count identifies the number of elements in the IntArray parameter.

IntArray is an array of integers that will receive the current column order.

Columns can be re-ordered by the user via drag and drop if the lvxHeaderDragDrop value is set in the ExtendedStyles property.

TExtListView.GetColumnOrder Example

TExtListView.GetColumnOrder

Reverse the current column order of a list view with four columns

```
var
  Columns: array[0..3] of integer;
  Tmp: integer;
begin
  // Get current order
  if AnExtListView.GetColumnOrder(4, Columns) then
  begin
    // Reverse the array values
     // Swap columns 1 and 4
    Tmp := Columns[0];
    Columns[0] := Columns[3];
    Columns[3] := Tmp;
     // Swap columns 2 and 3
    Tmp := Columns[1];
    Columns[1] := Columns[2];
    Columns[2] := Tmp;
     // Set the new order
    ExtListView.SetColumnOrder(4, Columns);
  end;
end;
```

TExtListView.GetColumnOrder See Also

ExtendedStyles
SetColumnOrder

GetSubItemAt Method

Overviews Example

Applies To
TExtListView

Declaration
```
function GetSubItemAt(X: integer; Y: integer): string;
```

Visibility
public

Description
GetSubItemAt returns the text of the Subitem or Caption value located at X, Y parmeter coordinates. The coordinates are relative to the control. If no text is at the given coordinates, an empty string is returned.

TExtListView.GetSubItemAt Example

TExtListView.GetSubItemAt

Displays Caption and Subitem text under the mouse in the status bar by attaching this code to the list view's OnMouseMove event

```
var
  SubItemText: string;
begin
  SubItemText := AnExtListView.GetSubItemAt(X, Y);
  if SubItemText <> '' then
    SubItemText := 'SubItem = ' + SubItemText;
  StatusBar.SimpleText := SubItemText;
end;
```

LoadSettings Method

Overviews See Also

Applies To
TExtListView

Declaration
```
function LoadSettings: boolean; override;
```

Visibility
public

Description
The LoadSettings method is used to load the width and order of all columns in the list view according to the values in the SaveSettings property.

This method is always called, even if the SaveSettings.AutoSave property is false. The values are simply not loaded in this case.

This method is called automatically when the component's window is created, so you should normally not need to call this method in your code.

TExtListView.LoadSettings See Also

StoreSettings
TExtLVSaveSettings

SetColumnOrder Method

Overviews Example See Also

Applies To
TExtListView

Declaration
```
procedure SetColumnOrder(Count: integer; const IntArray: array of integer);
```

Visibility
public

Description
SetColumnOrder sets the left-to-right order of columns in a list view control.

Count identifies the number of elements in the IntArray parameter.

IntArray is an array of integers that contains the new column order.

Columns can be re-ordered by the user via drag and drop if the lvxHeaderDragDrop value is set in the ExtendedStyles property.

TExtListView.SetColumnOrder Example

TExtListView.SetColumnOrder

Reverse the current column order of a list view with four columns

```
var
  Columns: array[0..3] of integer;
  Tmp: integer;
begin
  // Get current order
  if AnExtListView.GetColumnOrder(4, Columns) then
  begin
    // Reverse the array values
     // Swap columns 1 and 4
    Tmp := Columns[0];
    Columns[0] := Columns[3];
    Columns[3] := Tmp;
     // Swap columns 2 and 3
    Tmp := Columns[1];
    Columns[1] := Columns[2];
    Columns[2] := Tmp;
     // Set the new order
    ExtListView.SetColumnOrder(4, Columns);
  end;
end;
```

TExtListView.SetColumnOrder See Also

ExtendedStyles
GetColumnOrder

SetIconSpacing Method

Overviews

Applies To
TExtListView

Declaration
```
procedure SetIconSpacing(X: integer; Y: integer);
```

Visibility
public

Description
SetIconSpacing sts the spacing between icons when in vsIcon ViewStyle.

X is the distance in pixels to set between icons on the x-axis.

Y is the distance in pixels to set between icons on the y-axis.

SetItemCountEx Method

Overviews See Also

Applies To
TExtListView

Declaration
```
procedure SetItemCountEx(Count: integer; Flags: TLVItemCountFlags);
```

Visibility
public

Description
SetItemCountEx is used only in VirtualMode. It is used to set the number of items that the list view contains.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it. See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use.

TExtListView.SetItemCountEx See Also

OnVMCacheHint
OnVMFindItem
OnVMGetItemInfo
OnVMStateChanged
VirtualMode

StoreSettings Method

Overviews See Also

Applies To
TExtListView

Declaration
```
function StoreSettings: boolean; override;
```

Visibility
public

Description
The StoreSettings method is used to save all the values that are available. For TExtListView components, column widths and the column order are available.

This method is always called, even if the TEnhLVSaveSettings.AutoSave property is false. The values are simply not saved in this case.

This method is called automatically when the component's window is destroyed, so you should normally not need to call this method in your code.

TExtListView.StoreSettings See Also

LoadSettings
TEnhLVSaveSettings
TExtLVSaveSettings

UpdateColumnImage Method

Overviews Example See Also

Applies To
TExtListView

Declaration
```
procedure UpdateColumnImage(Index: integer);
```

Visibility
public

Description
UpdateColumnImage is used to refresh a specific column header image when using the ColumnsFormat property.

It is generally only needed internally, but I have included it because I found that changing column information (ColumnsFormat property) at run-time screws up the header images. For example, if you have set up the ColumnsFormat such that you have images appearing in your column headers and then you change the alignment of one of the TListColumns (an item of the Columns property), the header image will disappear. I could catch this if Borland would have been a little more liberal with their virtual/dynamic declarations, but the one I need isn't declared that way, so I can't override it. Instead, you will just have to be aware of the situation and manually refresh it yourself.

TExtListView.UpdateColumnImage Example

TExtListView.UpdateColumnImage

```
  // Change column 3's text alignment.
  ExtListView.Columns[2].Alignment := taRightJustify;
  // Screwed up header image, refresh it
  ExtListView.UpdateColumnImage(2);
```

TExtListView.UpdateColumnImage See Also

UpdateColumnsImages

UpdateColumnsImages Method

Overviews Example See Also

Applies To
TExtListView

Declaration
```
procedure UpdateColumnsImages;
```

Visibility
public

Description
UpdateColumnsImages is used to refresh all column header images when using the ColumnsFormat property.

It is generally only needed internally, but I have included it because I found that changing column information (ColumnsFormat property) at run-time screws up the header images. For example, if you have set up the ColumnsFormat such that you have images appearing in your column headers and then you change the alignment of one of the TListColumns (an item of the Columns property), the header image will disappear. I could catch this if Borland would have been a little more liberal with their virtual/dynamic declarations, but the one I need isn't declared that way, so I can't override it. Instead, you will just have to be aware of the situation and manually refresh it yourself.

TExtListView.UpdateColumnsImages Example

TExtListView.UpdateColumnsImages

```
  // Change column 3's text alignment.
  ExtListView.Columns[2].Alignment := taRightJustify;
  // Screwed up header image, refresh it
  ExtListView.UpdateColumnImage(2);
```

TExtListView.UpdateColumnsImages See Also

UpdateColumnImage

BackgroundImage Property
Overviews

Applies To
TExtListView

Declaration
```
property BackgroundImage: TELVBackgroundImage;
```

Visibility
published

Description
BackgroundImage is an experimental property that is not currently implemented because I can't get it to work. If you want to try to get it to work, enable the DFS_TRY_BACKGROUND_IMAGE define located at the top of ExtListView.pas. Let me know if you have any luck.

ColumnsFormat Property
Overviews

Applies To
TExtListView

Declaration
```
property ColumnsFormat: TExtListColumns;
```

Visibility
published

Description
ColumnsFormat describes the formatting to be applied to the list view column headers.   This formatting consists of an image index into the SmallImages image list property and the location of the image.   This information is represented by the TExtListColumns collection of TExtListColumn items.

ExtendedStyles Property

Overviews

Applies To
TExtListView

Declaration
```
property ExtendedStyles: TLVExtendedStyles;
```

Visibility
published

Description
The ExtendedStyles property is a set that identifies what extended styles of the COMCTL32.DLL update are to be used.

(***RPM*** = Report Mode Only)

· lvxGridlines:	Adds grid lines to seperate items and columns. ***RPM***
· lvxSubItemImages:	Allows images to be displayed for subitems. ***RPM***
· lvxCheckboxes:	Adds checkboxes to items. Checked items are stored internally as selected items.
· lvxTrackSelect:	Tracks the mouse and highlights the item it currently positioned over by changing it's color. If mouse is left over an item for a brief period of time, it will be automatically selected.
· lvxHeaderDragDrop:	Allows headers to be dragged to new positions and dropped, allowing users to reorder column information.
· lvxFullRowSelect:	Allows user to click anywhere on an item to select it, highlighting the entire length of the item. Without this style, users must click inside the text of column 0. It is only useful in vsReport view style.
· lvxOneClickActivate:	Sends an LVN_ITEMACTIVATE notification message to the parent when the user clicks an item.
· lvxTwoClickActivate:	Sends an LVN_ITEMACTIVATE notification message to the parent when the user double clicks an item.
· lvxFlatScrollBar:	Enables flat scroll bars in the list view.
· lvxUnderlineHot:	Causes hot items to be displayed with underlined text. This style is ignored if lvxOneClickActivate or lvxTwoClickActivate is not set.
· lvxUnderlineCold:	Causes non-hot items to be displayed with underlined text. This style is ignored if lvxOneClickActivate is not set.

HotCursor Property

Overviews See Also

Applies To

TExtListView

Declaration

```
property HotCursor: HCursor;
```

Visibility

public

Description

The HotCursor is the HCURSOR value used when the pointer is over an item while hot tracking is enabled.

Hot items are only used if one of the following ExtendedStyles values are set: lvxTrackSelect, lvxOneClickActivate, lvxTwoClickActivate. The appearance of hot and cold items are also controlled by the lvxUnderlineHot and lvxUnderlineCold styles.

TExtListView.HotCursor See Also

ExtendedStyles
HotItem

HotItem Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property HotItem: integer;
```

Visibility
public

Description
HotItem is the index of the list item that is considered to be the "hot" item.

Hot items are only used if one of the following ExtendedStyles values are set: lvxTrackSelect, lvxOneClickActivate, lvxTwoClickActivate. The appearance of hot and cold items are also controlled by the lvxUnderlineHot and lvxUnderlineCold styles.

Hot items were introduced in the v4.70 COMCTL32.DLL update and are generally defined as the item under the cursor. However, you can write to the property to set the hot item in code.

TExtListView.HotItem See Also

ExtendedStyles
HotCursor

HoverTime Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property HoverTime: DWORD;
```

Visibility
published

Description
HoverTime is the amount of time in milliseconds which the mouse cursor must hover over an item before it is selected. This only applies if the lvxTrackSelect, lvxOneClickActivate, or lvxTwoClickActivate value is set in the ExtendedStyles property.

TExtListView.HoverTime See Also

ExtendedStyles
OnHotTrack

IsChecked Property

Overviews

Applies To
TExtListView

Declaration
```
property IsChecked [Index: integer]: boolean;
```

Visibility
public

Description
Use IsChecked array property to get or set an item's checked state.

This property only applies if the lvxCheckboxes value of the ExtendedStyles property is set.

ItemIndent Property

Overviews

Applies To
TExtListView

Declaration
```
property ItemIndent [Index: integer]: integer;
```

Visibility
public

Description
ItemIndent is a run-time only array property that can be used to get or set the indent level of a list item.

Use the zero based item index as the array index.

The indent value is in units equal to the size of the SmallImages Width property.   For example, if the TImageList attached to SmallImages had a Width value of 16, setting an item's indentation to 2 would indent the item 32 pixels.   If there is no image list assigned to SmallImages, no indenting will occur.

Indenting is only supported for Caption text, not SubItem text.
Indenting is not supported by default owner drawing.

Items Property

Overviews See Also

Applies To

TExtListView

Declaration

```
property Items;
```

Visibility

published

Description

The Items property for a TExtListView is the same as for a normal TListView except when it is in VirtualMode. In virtual mode, no item data is stored in the list view, therefor none of the TListItem values in the Items property are valid. In virtual mode, all values referenced by the Items property are NIL.

TExtListView.Items See Also

VirtualMode

OnHotTrack Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property OnHotTrack: TLVHotTrackEvent;
```

Visibility
published

Description
The OnHotTrack event is fired the user moves the mouse over an item. This notification is only sent when the lvxTrackSelect value of the ExtendedStyles property is set.

ItemIndex is a variable parameter that indicates the item index being selected by hot tracking. Upon entering, this is the value of the item under the cursor. You can modify this value to change which item is to be "tracked".

SubItemIndex indicates which subitem the cursor is over.

Location is a TPoint value specifying the X, Y mouse coordinates relative to the control.

AllowSelect is a variable boolean parameter that can be set to FALSE to prevent the track selection.

TExtListView.OnHotTrack See Also

ExtendedStyles
HoverTime

OnInfoTip Property
Overviews

Applies To
TExtListView

Declaration
```
property OnInfoTip: TLVInfoTipEvent;
```

Visibility
published

Description
OnInfoTip is an experimental event that is not currently implemented because I can't get it to work.   If you want to try to get it to work, enable the DFS_TRY_INFOTIP define located at the top of ExtListView.pas. Let me know if you have any luck.

OnItemActivate Property

Overviews See Also

Applies To
TExtListView

Declaration

```
property OnItemActivate: TLVItemActivateEvent;
```

Visibility
published

Description
The OnItemActivate event is fired when an item or items are activated. When an item is activated is controlled by the ExtendedStyles property. If the lvxOneClickActivate value is set, the event fires when the user single clicks an item. If lvxTwoClickActivate is set, the event fires on double clicks. If niether is set, the event will **never** fire.

TExtListView.OnItemActivate See Also

ExtendedStyles

OnMarqueeBegin Property

Overviews

Applies To
TExtListView

Declaration
```
property OnMarqueeBegin: TLVMarqueeBeginEvent;
```

Visibility
published

Description
The OnMarqueeBegin event is fired when a bounding box (marquee) selection has begun. A bounding box selection is the process of clicking the list view window's client area and dragging to select multiple items simultaneously.

CanBegin is a variable parameter that you fill in to indicate whether the marquee selection should be allowed or not.

OnVMCacheHint Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property OnVMCacheHint: TLVVMCacheHintEvent;
```

Visibility
published

Description
The OnVMCacheHint event applies only when VirtualMode is enabled. This event is fired when the contents of a virtual list view control's display area have changed. For example, it will fire when the user scrolls the control's display. This is used to notify the application of which items the control thinks are mostly likely to be needed soon. In this way, the application can preload this data so that it is readily available when the OnVMGetItemInfo events begin firing.

HintInfo is a record containing information about the range of items to be cached. The iFrom and iTo members indicate the range of items that they control is suggesting that you cache.

Note that this event is not always an exact representation of the items that will be requested by OnVMGetItemInfo. Therefore, if the requested item is not cached while handling OnVMGetItemInfo, the application must be prepared to supply the requested information from a source outside the cache.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it. See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use. Note that only VMDemo implements a caching scheme via the OnVMCacheHint event.

TExtListView.OnVMCacheHint See Also

OnVMFindItem
OnVMGetItemInfo
OnVMStateChanged
SetItemCountEx
VirtualMode

OnVMFindItem Property

Overviews See Also

Applies To
TExtListView

Declaration

```
property OnVMFindItem: TLVVMFindItemEvent;
```

Visibility
published

Description
The OnVMFindItem event applies only when VirtualMode is enabled. This event fires when it needs the to find a particular item. For example, the control will fire this event when it receives shortcut keyboard input or when it receives an LVM_FINDITEM message.

FindInfo describes the item that is being looked for. It is a record that consists of an iStart parameter that indicates what index the search should start from and lvif which is a record that further describes the item. lvif is a record that consists of the following members:

· flags The type of search to perform. It can be one or more of the following values (use bitwise AND to test):

LVFI_PARAM Searches based on the lParam member. The lParam member of the matching item's LVITEM structure must match the lParam member of this structure. If this value is specified, all other values are ignored.
LVFI_PARTIAL Checks to see if the item text begins with the string pointed to by the psz member. This value implies use of LVFI_STRING.
LVFI_STRING Searches based on the item text. Unless additional values are specified, the item text of the matching item must exactly match the string pointed to by the psz member.
LVFI_WRAP Continues the search at the beginning if no match is found.
LVFI_NEARESTXY Finds the item nearest to the position specified in the pt member, in the direction specified by the vkDirection member.

· psz Address of a null-terminated string to compare with the item text if flags specifies LVFI_STRING or LVFI_PARTIAL.

· lParam Value to compare with the lParam member of a list view item's LVITEM structure if the flags member specifies LVFI_PARAM.

· pt TPoint structure that specifies the starting position to search from. This member is used only if LVFI_NEARESTXY is specified in the flags member.

· vkDirection Direction to search. This member contains the virtual key code of an arrow key that corresponds to the direction to search. This member is used only if LVFI_NEARESTXY is specified in the flags member.

Found is a variable parameter that you set to the item index of the matching item. If not found, set this parameter to -1.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it. See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use.

TExtListView.OnVMFindItem See Also

OnVMCacheHint
OnVMFindItem
OnVMGetItemInfo
OnVMStateChanged
SetItemCountEx
VirtualMode

OnVMGetItemInfo Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property OnVMGetItemInfo: TLVVMGetItemInfoEvent;
```

Visibility
published

Description
The OnVMGetItemInfo event applies only when VirtualMode is enabled. The event is fired when the list view needs information about an item it wants to display. Virtual mode list views do not store any of the information themselves, only the number of items and their selection states. All other information is provided to the list view via variable parameters of this event.

Sender is the TExtListView component requesting the information.

Item is the index (zero based) of the item that is needed.

SubItem is the index (one based) of the subitem being requested. If this parameter is zero, the request is for the item itself, the Caption text for example.

Mask is a set that may contain any of the following values:

·	lvifText	The text for this item is needed, assign a value to the Text parameter.
·	lvifImage	The image index is needed, assign a value to the Image parameter.
·	lvifParam	The lParam value for the item is needed, assign a value to the Param parameter. TExtListView does not directly use this for anything but it relates to the LVM_FINDITEM and LVM_SORTITEMS messages, so you may need to respond to this if you use either of those messages.
·	lvifState	The state of the item is being requested, assign a value to the State parameter. I am unsure of the real use of this. The selection state of items is stored by the list view in virtual mode, so I'm note sure what you would need this for. Perhaps other states are not stored, but I'm not sure.
·	lvifIndent	The indent level of the item is being requested, assign a value to the Indent parameter.

These variable parameters only need to be filled in if their corresponding flag is set in the Mask parameter (see above).

Image is the zero based image index of the item or subitem.

Param is a 32 bit application defined value.

State can be a combination of any of the LVIS_xxx flags bitwise ORed together.

Indent is the number of units to indent the item (one unit equals the width of
the SmallImages image list).

Text is the string representing the item or subitem.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it.   See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use.

TExtListView.OnVMGetItemInfo See Also

OnVMCacheHint
OnVMFindItem
OnVMStateChanged
SetItemCountEx
VirtualMode

OnVMStateChanged Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property OnVMStateChanged: TLVVMStateChangedEvent;
```

Visibility
published

Description
The OnVMStateChanged event applies only when VirtualMode is enabled. According to Microsoft's documentation, this event should fire when the state of an item or range of items has changed. In practice, however, I have never seen this event fire. I have never seen the notification message that drives this event being sent. As best as I can tell, it appears not to have been implemented and I would suggest that you not rely on it even if it does appear to work in your system.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it. See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use.

TExtListView.OnVMStateChanged See Also

OnVMCacheHint
OnVMFindItem
OnVMGetItemInfo
SetItemCountEx
VirtualMode

RequireComCtlUpdate Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property RequireComCtlUpdate: boolean;
```

Visibility
published

Description
The RequireComCtlUpdate property indicates whether the COMCTL32.DLL on the user's system must be v4.70 or higher. If this property is true, and the DLL is not at least that version, an EELVOldComCtl exception will be raised.

Generally, you will want to catch the EELVOldComCtl exception in your Application object's OnException event handler and provide a nice, friendly message for the user.

Except for VirtualMode, the TExtListView component will have no problem running on a system with a prior version of COMCTL32.DLL, it simply will not the new features available. For example, if you specified lvxFullRowSelect in ExtendedStyles, the component would simply function as normal with only the Caption text selectable. Using any of the extended functionality would essentially be the same as no operation. All functionality that is inherited from TEnhListView would still be available and operate as expected. As mentioned, VirtualMode would not work at all because no data is ever stored in the list view; the list view would simply be empty.

TExtListView.RequireComCtlUpdate See Also

CheckComCtlVersion

SaveSettings Property

Overviews

Applies To

TExtListView

Declaration

```
property SaveSettings: TExtLVSaveSettings;
```

Visibility

published

Description

SaveSettings is used to describe what user settings should be stored and retrieved automatcially between sessions.   See the TExtLVSaveSettings class and it's ancestor TEnhLVSaveSettings for a complete description.

SmallImages Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property SmallImages: TImageList;
```

Visibility
published

Description
SmallImages provides a list of icon images to display for each item in the list when ViewStyle is not vsIcon. The images are also used for the column header images if the ColumnsFormat property is being used.

Set SmallImages to specify the icons that should be displayed next to text in the column headers and items in the list when ViewStyle is vsSmallIcon, vsList, or vsReport. Each item in the Items list can be associated with an icon in this image list by setting its ImageIndex property, as can each column by using the ColumnsFormat property.

TExtListView.SmallImages See Also

ColumnsFormat

SubItem_BoundsRect Property

Overviews See Also

Applies To
TExtListView

Declaration
`property SubItem_BoundsRect [Item: integer; SubItem: integer]: TRect;`
Read only property

Visibility
public

Description
SubItem_BoundsRect is a read-only array property that retrieves a TRect describing the bounding rectangle of the entire item, including the icon and label. This property is intended to be used only with list view controls in the vsReport ViewStyle mode.

Item is the zero-based index of the subitem's parent item.

SubItem is the one-based index of the subitem.

TExtListView.SubItem_BoundsRect See Also

SubItem_IconRect
SubItem_LabelRect
SubItem_SelectBoundsRect

SubItem_IconRect Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property SubItem_IconRect [Item: integer; SubItem: integer]: TRect;
```
Read only property

Visibility
public

Description
SubItem_IconRect is a read-only array property that retrieves a TRect describing the bounding rectangle of the icon or small icon. This property is intended to be used only with list view controls in the vsReport ViewStyle mode.

Item is the zero-based index of the subitem's parent item.

SubItem is the one-based index of the subitem.

TExtListView.SubItem_IconRect See Also

SubItem_BoundsRect
SubItem_LabelRect
SubItem_SelectBoundsRect

SubItem_ImageIndex Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property SubItem_ImageIndex [Item: integer; SubItem: integer]: integer;
```

Visibility
public

Description
SubItem_ImageIndex is an array property used to get and set the image index for subitems. Subitems can only have images if the lvxSubItemImages value is set in ExtendedStyles.

Note: Setting a subitem image to a value of -1 does not properly clear the image for the subitem as it does for TListItem.ImageIndex. The current COMCTL32.DLL implementation does not seem to store this value and instead it gets a random value assigned to it. The work-around that I have found is to set the index to a value that does not exist in the image list. To make this a bit easier, I have declared a constant named **ELV_NO_SUBITEM_IMAGE** that is equal to MaxInt-1 (MaxInt seems to suffer the same problem as -1). This will work fine as long as your image list does not have more than 2,147,483,646 images in it. :)

TExtListView.SubItem_ImageIndex See Also

ExtendedStyles

SubItem_LabelRect Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property SubItem_LabelRect [Item: integer; SubItem: integer]: TRect;
```
Read only property

Visibility
public

Description
SubItem_LabelRect is a read-only array property that retrieves a TRect describing the bounding rectangle of the entire item, including the icon and label. This property is intended to be used only with list view controls in the vsReport ViewStyle mode.

Item is the zero-based index of the subitem's parent item.

SubItem is the one-based index of the subitem.

TExtListView.SubItem_LabelRect See Also

SubItem_BoundsRect
SubItem_IconRect
SubItem_SelectBoundsRect

SubItem_SelectBoundsRect Property

Overviews See Also

Applies To
TExtListView

Declaration
```
property SubItem_SelectBoundsRect [Item: integer; SubItem: integer]: TRect;
```
Read only property

Visibility
public

Description
SubItem_SelectBoundsRect is a read-only array property that retrieves a TRect describing the selection rectangle for a subitem in a list view control. This property is intended to be used only with list view controls in the vsReport ViewStyle mode.

Item is the zero-based index of the subitem's parent item.

SubItem is the one-based index of the subitem.

TExtListView.SubItem_SelectBoundsRect See Also

SubItem_BoundsRect
SubItem_IconRect
SubItem_LabelRect

VirtualMode Property

Overviews See Also

Applies To
TExtListView

Declaration

```
property VirtualMode: boolean;
```

Visibility
published

Description
VirtualMode indicates whether teh list view should operate in virtual mode. In virtual mode, a list view does not store any information about the items it contains except for the selection state. All other data is supplied by the application via the OnVMGetItemInfo event.

To set the number of items a virtual list view contains, use the SetItemCountEx method.

Because no item data is stored in the component, the Items property is not used at all. Attempting to reference an TListItem will always result in a NIL pointer.

A complete discussion of virtual mode is beyond this help file, but two demonstration programs are included with this component that illustrate how to use it. See the VMDemo application for a (relatively) simplified implementation, and TestVM for a "real world" use.

For a complete discussion of virtual mode, you should refer to Microsoft's documentation at http://www.microsoft.com/msdn/sdk/inetsdk/help/itt/CommCtls/ListView/
Updates.htm#sec_listview_updates

TExtListView.VirtualMode See Also

OnVMCacheHint
OnVMFindItem
OnVMGetItemInfo
OnVMStateChanged
SetItemCountEx

WorkArea Property
Overviews

Applies To
TExtListView

Declaration
```
property WorkArea: TRect;
```
Write only property

Visibility
public

Description
WorkArea is a write-only property that appears to have been dropped from the latest version (4.72) of COMCTL32.DLL. It likely has been replaced by the multiple work area support that was added in v4.71.

TExtListView does not yet support this new multiple work area functionality. If you need it, you will have to use the ListView_GetWorkAreas and ListView_SetWorkAreas functions (or LVM_GETWORKAREAS and LVM_SETWORKAREAS messages) directly.

TExtLVSaveSettings Class

Methods Properties Hierarchy Overviews

Ancestor
TEnhLVSaveSettings

Unit
ExtListView

Description
The TExtLVSaveSettings class is used to describe what elements of a TExtListView to automatically save and restore, if any, and where to save those values in the registry.

TExtLVSaveSettings Methods

- Create
- StoreColumnOrder
- ReadColumnOrder

TExtLVSaveSettings Properties

- SaveColumnOrder

TExtLVSaveSettings Hierarchy

TPersistent

- TEnhLVSaveSettings
  - TExtLVSaveSettings

Create Method

Overviews

Applies To

TExtLVSaveSettings

Declaration

```
constructor Create; override;
```

Visibility

public

Description

The Create constructor creates a new instance of the class and initializes internal variables.   It does nothing out of the ordinary that you should be concerned with.

ReadColumnOrder Method
Overviews

Applies To
TExtLVSaveSettings

Declaration
```
procedure ReadColumnOrder(ColCount: integer; var IntArray: array of integer);
```

Visibility
public

Description
ReadColumnOrder is the method used to actually read the column order values from the registry. ColCount is the number of elements that the IntArray parameter can hold, and IntArray will be populated with the column widths. The values are read from the registry key indicated by the TEnhLVSaveSettings.RegistryKey property.

StoreColumnOrder Method
Overviews

Applies To
TExtLVSaveSettings

Declaration
```
procedure StoreColumnOrder(ColCount: integer; const IntArray: array of
integer);
```

Visibility
public

Description
StoreColumnOrder is a method used to actually write the column order values to the registry.   ColCount is the number of elements being passed in the IntArray parameter, which holds the actual column order. The values are stored in the registry key indicated by the TEnhLVSaveSettings.RegistryKey property.

SaveColumnOrder Property

Overviews See Also

Applies To
TExtLVSaveSettings

Declaration
```
property SaveColumnOrder: boolean;
```

Visibility
published

Description
SaveColumnOrder indicates whether the column order of the list view should be saved and restored between sessions.

TExtLVSaveSettings.SaveColumnOrder See Also

TEnhLVSaveSettings.AutoSave