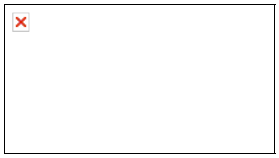
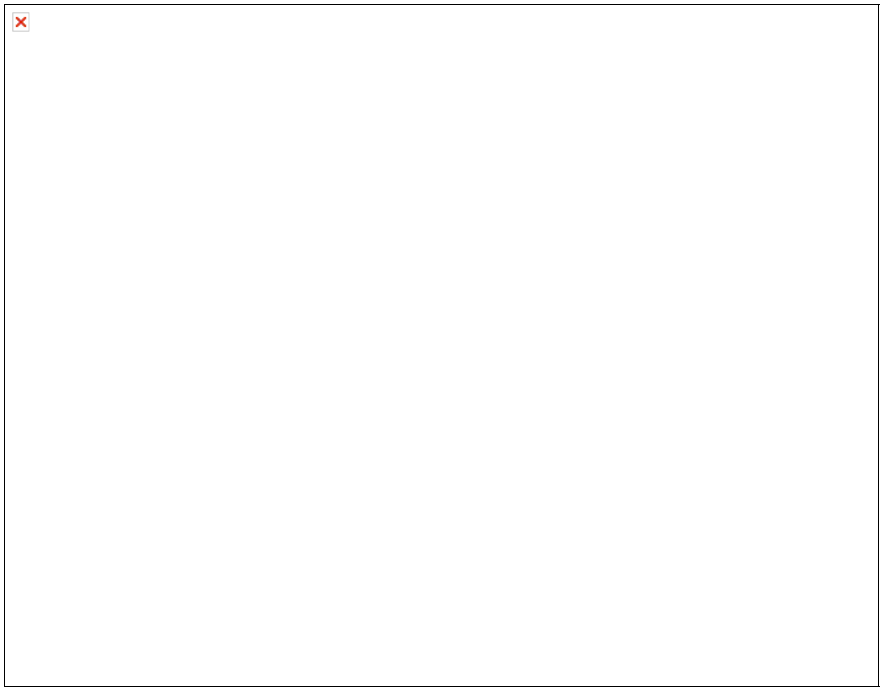
CTreeListCtrl

class CTreeListCtrl : public CTreeCtrl



Include file: TreeListCtrl.h
Source files: TreeListCtrl.cpp TreeListWnd.cpp

The class CTreeListCtrl code is an implementation of a tree view, combined with a list view. The view is compatible to the tree control of the common controls (see CTreeCtrl). It allows the user to set colors, text, and icons for each item separately. An MFC class for the view is also implemented.



Features

- The window messages are compatible with the tree control of the common controls.
- Alternating colors are provided for each line.
- The color, style, and icon could be set for each item separately. (see here)
- An index access is implemented so that you can use the control as ListView. (see here)
- It is possible to add an user data area to each item. (see here)
- To each column an auto edit feature can be assigned, so no callback function must be used (see here)

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members

CTreeListCtrl Class Members

- Construction
- Attributes
- Sort functions
- Operations
- Column functions
- User data functions
- List view functions
- Notify messages

Construction

CTreeListCtrl	Constructs a **CTreeListCtrl** object.
CreateEx	Creates a tree list view control with an extended style.
Create	Creates a tree list view control and attaches it to a **CTreeListCtrl** object.

Attributes

EnumItems	Enums all items of as tree list control.
GetChildItem	Retrieves the first child item of an entry.
GetCount	Retrieves the number of tree items associated with a tree view control.
GetColor	Retrieves the color of a specified display element.
GetComboControl	Gets the current combo box.
GetDropHilightColumn	Retrieves the column which contains the drop highlighted item.
GetExtendedStyle	Retrieves the extended style flags.
GetEditControl	Gets the current edit window.
GetFirstItem	Retrieves the first item of the control.
GetFirstSelected	Retrieves the first selected item.
GetFirstVisibleItem	Retrieves the first visible item.
GetFocusColumn	Retrieves the column which contains the focus item.
GetFocusItem	Retrieves the item handle which has the focus.
GetImageList	Retrieves the image list a control.
GetItemBkColor	Retrieves the background color of an item.
GetItemCheckBox	Gets the button state.
GetItemImage	Retrieves the images of an item.
GetItemImageEx	Retrieves the image of an item, with a column parameter.
GetItemOfRow	Retrieves the handle of a item from a row.
GetItemRect	Retrieves the bounding rectangle of a item.
GetItemState	Retrieves the state of an item.
GetItemText	Retrieves the text of an item.
GetItemTextColor	Retrieves the text color of an item.
GetLastChild	Retrieves the last child item.
GetNextItem	Retrieves a item with a specified relationship.
GetNextSelected	Retrieves the next selected item.
GetNextSelectedChild	Retrieves the next selected child item of a parent.
GetNextSiblingItem	Retrieves the next sibling item.
GetNextVisibleItem	Retrieves the next visible item.
GetParentItem	Retrieves the parent item of an entry.
GetPrevSiblingItem	Retrieves the previous sibling item.
GetPrevVisibleItem	Retrieves the previous visible item.
GetRootItem	Retrieves the handle of the root item
GetRowCount	Retrieves the total count of rows in the tree list control.
GetRowOfItem	Retrieves the row of an item.
GetCountPerPage	Retrieves the count of visible rows.
GetSelectionColumn	Retrieves the count of selected items.
GetStyle	Retrieves the style flags of the control.
IsItemVisible	Checks if an item is visible.
SetColor	Sets the color of a specified display element.
SetExtendedStyle	Sets some extended style flags the control.
SetFocusItem	Selects the item with the focus.
SetItemBkColor	Sets the background color of an item.
SetItemCheckBox	Sets the button state.
SetItemState	Changes the state of an item.
SetItemText	Changes the text of an item.
SetItemTextColor	Changes the text color of an item.
SetStyle	Sets some style flags of the control.

Column functions

DeleteColumn	Deletes a column from the header control.
DisableItemAutoEdit	Disables the column auto edit option.
FixColumnSize	Fixes the size of a a column.
GetColumn	Retrieves the attributes of a tree list control's column.
GetColumnCount	Get the count of columns.
GetColumnOrderArray	Gets an array with the column order.
GetColumnWidth	Get the width of a column in pixels.
GetFocusColumn	Get the number of the column which has the focus.
GetHeaderCtrl	Gets pointer to the header control.
InsertColumn	Inserts a column in the header of the Tree-List-Control
SetColumn	Sets the attributes of a tree list control's column.
SetColumnAutoEdit	Sets the auto edit mode for a column.
SetColumnAutoIcon	Sets the auto icon mode for a column.
SetColumnImage	Sets the text of an item in the column header
SetColumnMark	Sets the mark state of a column.
SetColumnOrderArray	Sets the array with the column order.
SetColumnText	Sets the text of an item in the column header
SetColumnWidth	Set the width of a column in pixels.

Sort functions

SortChildren	Sorts the children of a given parent item.
SortChildrenCB	Sorts the children of a given parent item using an application-defined sort function.
SortChildrenEX	Sorts the children of a given parent item using an application-defined extended sort function.

User data functions

GetUserData	Retrieves the pointer to the user data of an item.
GetUserDataSize	Retrieves the user data size.
SetUserDataSize	Changes the user data size.

Operations

CreateDragImage	Creates a dragging bitmap for the given item in a tree list control.
CollapseAll	Collpses an item and his childs.
DeleteChildItems	Deletes all childs of an item.
DeleteItem	Deletes an items in a tree view control.
DeleteAllItems	Deletes all items in a tree view control.
Expand	Expands, or collapses, the child items of the specified tree view item.
ExpandAll	Expands all parent items of an entry.
EditLabel	Edits a specified tree view item in-place via edit box.
EditLabelCb	Edits a specified tree view item in-place via combo box.
EnsureVisible	Ensures that a tree list item is visible in its tree list control.
FindItem	Searches for an item in the control with several properties.
InsertItem	Inserts a new item in a tree view control.
HitTest	Returns the current position of the cursor related to the CTreeCtrl object.
Select	Selects, scrolls into view, or redraws a specified tree view item.
SelectChilds	Selects or deselects all childs of an item.
SelectDropTarget	Redraws the tree item as the target of a drag-and-drop operation.
SelectItem	Selects a specified tree view item.
SelectSetFirstVisible	Selects a specified tree view item as the first visible item.
SetImageList	Sets the image lists of the control.
SetItemImage	Changes the images of an item.
SetItemImageEx	Changes the image of an item, with a column parameter.
SetItem	Changes some properties of an item.
StartEdit	Starts a label edit via the TVN_STARTEDIT notify message.

List view functions

ListCreateDragImage	Creates a dragging bitmap for the given item in a tree list control.
ListDeleteItem	Deletes an item in a Tree-List-Control which is used as List-Control
ListEditLabel	Edits a specified tree view item in-place via edit box.
ListEditLabelCb	Edits a specified tree view item in-place via combo box.
ListEnsureVisible	Ensures that a tree list item is visible in its tree list control.
ListGetColor	Gets the colors of an item in a Tree-List-Control which is used as List-Control
ListGetFirstSelected	Retrieves the row of the first selected item.
ListGetFocusItem	Retrieves the row of the item which has the focus.
ListGetItemBkColor	Retrieves the background color of an item.
ListGetItemCheckBox	Gets the checkbox state from an item in a row.
ListGetItemImage	Retrieves the image of an item.
ListGetItemImageEx	Retrieves the image of an item with a column parameter.
ListGetItemRect	Retrieves the bounding rectangle of an item.
ListGetItemState	Retrieves the state flags of an item.
ListGetItemText	Retrieves the item text.
ListGetItemTextColor	Retrieves the text color of an item.
ListGetNextSelected	Retrieves the next selected item.
ListGetTopIndex	Returns the index of the topmost visible item.
ListGetUserData	Retrieves the pointer to the user data of an item.
ListInsertItem	Inserts an item in a tree list control which is used as List-Control
ListSelectDropTarget	Redraws the tree item as the target of a drag-and-drop operation.
ListSelectItem	Selects an item item.
ListSetColor	Changes the colors of an item.
ListSetFocusItem	Selects the item with the focus.
ListSetItem	Changes an item in a tree list control.
ListSetItemBkColor	Sets the background color of an item.
ListSetItemCheckBox	Sets the checkbox state of an item in a row.
ListSetItemImage	Sets the image of an item.
ListSetItemImageEx	Sets the image of an item with a column parameter.
ListSetItemState	Sets the state flags of an item.
ListSetItemText	Sets the text of an item.
ListSetItemTextColor	Sets the text color of an item.
ListSetTopIndex	Scrolls to the item specified by an index at the top of the view.

Notify messages

TVN_CBSTATECHANGED	A combo box in a auto edit column was changed.
TVN_COLUMNCHANGED	The size of a column was changed.
TVN_ENDLABELEDIT	A label edit was finished.
TVN_ITEMTOOLTIP	Used to show an user defined tooltip.
TVN_STARTEDIT	Send if a lebel edit acion could be taken.
TVN_STEPSTATECHANGED	A step state in a auto edit column was changed.
CollapseAll	Text

see also: CTreeListCtrl

CTreeListCtrl::CTreeListCtrl

CTreeListCtrl ( )

Is the constructort of the class.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | Extended Styles | Styles

CTreeListCtrl::Create

BOOL Create (DWORD dwStyle , const RECT &sRect , CWnd *pParentWnd , UINT nId)
BOOL Create (DWORD dwStyle , UINT uItem , CDialog *pDlg , UINT nId)

Creates a new Tree-List-Control.

dwStyle	Specifies the tree view control's style. Apply any combination of tree view control styles to the control.
sRect	Specifies the tree view control's size and position. It can be either a CRect object or aRECT structure.
uItem	Specifies the id of a dialog item, over that the new window will be placed.
pDlg	Pointer to parent dialog.
pParentWnd	Specifies the tree view control's parent window, usually a CDialog. It must not be NULL.
nId	Specifies the tree view control's ID.

Returns nonzero if initialization was successful, otherwise 0.

If you specify the tree control in a dialog box template, or if you are using CTreeListCtrl, your tree control is created automatically when the dialog box or view is created. If you want to create the tree control as a child window of some other window, use the **Create** member function. If you create the tree control using **Create**, you must pass it **WS_VISIBLE**, in addition to other tree view styles.

You construct a **CTreeCtrl** in two steps. First call the constructor, then call **Create**, which creates the tree view control and attaches it to the **CTreeCtrl** object.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Extended Styles | GetExtendedStyle | GetStyle | SetExtendedStyle | SetStyle | CreateEx

CTreeListCtrl::CreateEx

BOOL CreateEx (DWORD dwExStyle , DWORD dwStyle , const RECT &sRect , CWnd *pParentWnd , UINT nId)
BOOL CreateEx (DWORD dwExStyle , DWORD dwStyle , UINT uItem , CDialog *pDlg , UINT nId)

Creates a new Tree-List-Control

dwExStyle	Is the exdended style of the window (see at WS_EX_????)
dwStyle	Specifies the tree view control's style. Apply any combination of tree view control styles to the control.
sRect	Specifies the tree view control's size and position. It can be either a CRect object or aRECT structure.
uItem	Specifies the id of a dialog item, over that the new window will be placed.
pDlg	Pointer to parent dialog.
pParentWnd	Specifies the tree view control's parent window, usually a CDialog. It must not be NULL.
nId	Specifies the tree view control's ID.

Returns nonzero if initialization was successful, otherwise 0.

If you specify the tree control in a dialog box template, or if you are using CTreeListCtrl, your tree control is created automatically when the dialog box or view is created. If you want to create the tree control as a child window of some other window, use the **Create** member function. If you create the tree control using **Create**, you must pass it **WS_VISIBLE**, in addition to other tree view styles.

You construct a **CTreeCtrl** in two steps. First call the constructor, then call **Create**, which creates the tree view control and attaches it to the **CTreeCtrl** object.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Extended Styles | GetExtendedStyle | GetStyle | SetExtendedStyle | SetStyle | Create

CTreeListCtrl Extended Styles

TVS_EX_ALTERNATECOLOR Draws each row of the in an alternating color.

TVS_EX_AUTOEXPANDICON When this flag is set and an entry in the control is expanded and has childs the image index will be increased by one. So it is possible to implement a simple open icon functionality without notify messages. It is only nessosary to place the open icon at the following position of the common icon. Use the flag **TV_NOAUTOEXPAND** with the icon number, to disable this function, for a single icon.

TVS_EX_AUTOHSCROLL Scrolls automatically to the column of the selected item. The horizontal scorollbar will hidden.

TVS_EX_BITCHECKBOX If this style is selected and **TVS_CHECKBOXES**, a click on the checkbox of the item only inverts the first bit of the **TVIS_STATEIMAGEMASK**. If **TVS_EX_BITCHECKBOX** isn't selected, a click on the checkbox switches between 0x1000 and 0x2000. With this style it is possible to display several checkbox styles. Insert state images with several checkbox pairs. The first bit (0x1000) is the checked state, the next 3 bits (0xE000) represents the checkbox style.

TVS_EX_EDITCLICK This option allowes to edit an selected item with a single mouse click. The selected column must be an auto edit column.

TVS_EX_FIXEDCOLSIZE If this flag is enabled, and the user increases a column width, automaticaly the width of the next column will be decreased. So the width of all columns remains constand.

TVS_EX_FULLROWITEMS This option allows to draw the items over the full raw of the window. That means that the left side of an item will be drawn in the item color instead of the background color. This flag should be used with **TVS_EX_ALTERNATECOLOR** and **TVS_EX_ITEMLINES**.

TVS_EX_FULLROWMARK This flag enables the selection of the full row. The selection mark will begin at the item icon in the first column, and ends after the last column.

TVS_EX_GRAYEDDISABLE Draw the control gray, if it is disabled.

TVS_EX_HEADERCHGNOTIFY If this style is set, a notify massege (**TVN_COLUMNCHANGED**)will be send to the parent window if the size of a comumn will be changed..

TVS_EX_HEADERDRAGDROP Allowes to sort the order of sub columns via darg and drop in the header.

TVS_EX_HIDEHEADERS Don't show the header control in the tree list window.

TVS_EX_HOMEENDSELECT Against the CTreeCtrl you can select the first and last item with the Cltr+Home or Ctrl+End keys.

TVS_EX_ITEMLINES With this style a frame is drawn around the item bodys.

TVS_EX_MULTISELECT This flag allows to select more than one item at same time. (see GetFirstSelectedItem)

TVS_EX_NOCHARSELCET This flag disables the selection of a single row entrys via keyboard input. i.e.: The input of 'D' selects the next entry which begins with a 'D' char.

TVS_EX_NOCURSORSET Don't set the cursor to the click point, if **TVS_EX_EDITCLICK** is used.

TVS_EX_NOCOLUMNRESIZE The user can't change the column size.

TVS_EX_SHAREIMAGELISTS Don't destroy the attached image lists, if the control is closed.

TVS_EX_SINGLECHECKBOX Allows to select only one checkbox in the first column.

TVS_EX_STEPOUT If this flags is set, the user can leave an edit control with the cursor buttons.

TVS_EX_SUBSELECT This flag allows to select each entry in each column. Without this flag only the entry in the first colum could be selected.

TVS_EX_TOOLTIPNOTIFY If this style is set, a notify massege (**TVN_ITEMTOOLTIP**)will be send to the parent window if the mouse cursor moves over an item. In the response of the massege the parent can change the text, position, and the delay to showing the tooltip window.

```
ON_NOTIFY(TVN_ITEMTOOLTIP,IDC_TREELIST,OnTooltipNotify)

...

void CMyDialog::OnTooltipNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
NM_TREEVIEW *pNmTreeView = (NM_TREEVIEW*)pNmHdr;
                                                  // Is the text for the tootip (""=no Tooltip)
   pNmTreeView->itemNew.pszText= "User definded\nTooltip\nMessage";
   pNmTreeView->ptDrag.x      += 20;              // Move Tooltip to an other position
   pNmTreeView->ptDrag.y      += 20;
   pNmTreeView->itemNew.mask  |= TVIF_TOOLTIPTIME; // Set a delay to show the tooltip
   pNmTreeView->itemNew.lParam = 1000;

   *pResult = 1;                                  // 0=Common-Tooltip 1=User-Tooltip

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | GetExtendedStyle | GetStyle | SetExtendedStyle | SetStyle | Styles

CTreeListCtrl Styles

The following styles can be applied to a tree view control:

- **TVS_HASLINES** The tree view control has lines linking child items to their corresponding parent items.
- **TVS_LINESATROOT** The tree view control has lines linking child items to the root of the hierarchy.
- **TVS_HASBUTTONS** The tree view control adds a button to the left of each parent item.
- **TVS_EDITLABELS** The tree view control allows the user to edit the labels of tree view items.
- **TVS_CHECKBOXES** The tree view control has checkboxes before the icons. The state of the checkbox is controlled via the state-image-mask.
- **TVS_SHOWSELALWAYS** Causes a selected item to remain selected when the tree-view control loses focus.
- **TVS_DISABLEDRAGDROP** The tree-view control is prevented from sending **TVN_BEGINDRAG** notification messages.
- **TVS_FULLROWSELECT** Enables full-row selection in the tree view. The entire row of the selected item is highlighted, and clicking anywhere on an item's row will cause it to be selected. This style cannot be used in conjunction with the **TVS_HASLINES** style.
- **TVS_INFOTIP** The tree view control will send the **TVN_GETINFOTIP** notification to obtain tooltip information.
- **TVS_NONEVENHEIGHT** The height of the items can be set to an odd height with the **TVM_SETITEMHEIGHT** message. By default, the height of items must be an even value.
- **TVS_NOTOOLTIPS** The tree view control uses no tooltips.
- **TVS_TRACKSELECT** Enables hot tracking in a tree view control.
- **TVS_SINGLEEXPAND** When this style is enabled, changing the selection in the tree view will automatically cause the item being selected to expand and the item being unselected to collapse. If the mouse is used to single-click the selected item and that item is closed, it will be expanded. If the selected item is single-clicked when it is open, it will be collapsed.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | Extended Styles | GetExtendedStyle | GetStyle | SetExtendedStyle | SetStyle

CTreeListCtrl::EnumItems

HTREEITEM EnumItems (HTREEITEM hLast)

Enums all items in the tree view control.

hLast	Handle of the last items. To start a new enumeration set this parameter to **TVI_ROOT**.

Returns a handle to the next item, on NULL if no other item was found.

```
hItem = TVI_ROOT;

for(;;)
    {
    hItem = pTreeView->EnumItems(hItem);
    if(!hItem)break;
    ...
    }
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstItem | GetNextItem | GetNextSiblingItem

CTreeListCtrl::GetChildItem

HTREEITEM GetChildItem(HTREEITEM hParrent)

Gets the first child od an item.

hParrent	Is the handle of the parent.

Retrieves the handle of the child item if successful; otherwise NULL..

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetNextVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetFirstSelected | GetNextItem | GetNextSiblingItem | GetPrevSiblingItem | GetParentItem

CTreeListCtrl::GetComboControl

void GetComboControl (HWND &hWnd) const
CComboBox *GetComboControl ( ) const

Gets a pointer ot the handle to the current combo box window.

hWnd	Here the window handle of the combobox will be saved.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetEditControl

CTreeListCtrl::GetCount

UINT GetCount ( )

Returns the number of items in the tree view control, otherwise – 1.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemOfRow | GetRowCount | GetRowOfItem

CTreeListCtrl::GetCountPerPage

int GetCountPerPage ( ) const

Retrieves the count of visible rows in the view. Half rows will be counted to.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetRowCount | GetRowOfItem

CTreeListCtrl::GetColor

COLORREF GetColor (int iIndex) const

Retrieves the color of a specified display element.

iIndex	Is the index of the display element:

- **TVC_BK** background color.
- **TVC_ODD** is the odd background color if TVS_EX_ALTERNATECOLOR is used.
- **TVC_EVEN** is the even background color if TVS_EX_ALTERNATECOLOR is used.
- **TVC_FRAME** is the color of the frame lines (with TVS_EX_ITEMLINES)
- **TVC_TEXT** is the common text color
- **TVC_LINE** is the color of the lines around the buttons
- **TVC_BOX** is the inner color of the buttons
- **TVC_TRACK** is the color of draged items
- **TVC_MARK** is the color of a selected row
- **TVC_MARK_ODD** is the color of a selected odd row
- **TVC_MARK_EVEN** is the color of a selected even row
- **TVC_INSERT** is the color of the insert mark

Returns the specified color in the COLORREF format. A **TV_NOCOLOR** return value means the default color.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | SetColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::GetDropHilightColumn

int GetDropHilightColumn ( )

Call this function to retrieve the column which contains the item that is the target of a drag-and-drop operation.

Returns the number of the column.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | CreateDragImage | SelectDropTarget

CTreeListCtrl::GetEditControl

void GetEditControl (HWND &hWnd) const
CEdit *GetEditControl ( ) const

Gets a pointer ot the handle to the current edit window.

hWnd	Here the window handle of the edit control will be saved.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetComboControl

CTreeListCtrl::GetExtendedStyle

DWORD GetExtendedStyle ( )

Returns the extended style flags of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | GetStyle | SetExtendedStyle | SetStyle | Styles

CTreeListCtrl::GetFirstItem

HTREEITEM GetFirstItem ( )

Retrieves the first item of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFirstVisibleItem | GetNextItem | GetNextSiblingItem | GetNextVisibleItem

CTreeListCtrl::GetFirstSelected

HTREEITEM GetFirstSelected ( )

Retrieves the handle of the first selected item, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | SelectItem | Select

CTreeListCtrl::GetFirstVisibleItem

HTREEITEM GetFirstVisibleItem( )

Retrieves the handle of the first visible item, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetNextVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetFirstSelected

CTreeListCtrl::GetFocusColumn

INT GetFocusColumn ( )

Retrieves the number of the column which contains the focus item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | SelectItem

CTreeListCtrl::GetFocusItem

HTREEITEM GetFocusItem ( )

Retrieves the handle of the item which has the focus, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | SelectItem

CTreeListCtrl::GetImageList

CImageList *GetImageList (int nImageListType)

Call this function to get a image list from the control.

nImageListType	Type of image list to get. The image list can be one of the following values:

TVSIL_NORMAL	Sets the normal image list, which contains the selected and nonselected images for the tree view item.
TVSIL_STATE	Sets the state image list, which contains the images for tree view items that are in a user-defined state.
TVSIL_CHECK	Sets the image list, which contains the ckeckboxes for the auto-column-edit-mode.
TVSIL_SUBIMAGES	Sets the image list, for the sub columns (2...n). If this list is zero the normal image list is used for the columns. If in the main column an image index above 0x40000000 is defined, this image list is also be used.

Returns the pointer to the image list if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItemImageEx | SetImageList | GetItemImage | SetItemImage

CTreeListCtrl::GetItemBkColor

COLORREF GetItemBkColor (HTREEITEM hItem , int nCol = 0) const

Retrieves the background color of an item.

hItem	Handle of the item whose background color is to be retrieved.
nCol	Selects the column of the item raw.

Returns the RGB value of the background color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColor | GetItemTextColor | SetColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::GetItemCheckBox

INT GetItemCheckBox (HTREEITEM hItem , int nCol = 0 , UINT uMask = 0x0F)

Gets the TVIS_STATEIMAGEMASK bits from an item, with the button state.

hItem	Is the handle of the Item.
nCol	Is the colomn of the button.
uMask	Selects wich bits in the TVIS_STATEIMAGEMASK should be changed.

Returns the button state:

If TVS_EX_BITCHECKBOX isn't enabeld:
0 = not visible
1 = not selected
2 = selected
If TVS_EX_BITCHECKBOX is enabeld:
0 = not selected
1 = selected

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | SetItemState | SetItemCheckBox

CTreeListCtrl::GetItemImage

BOOL GetItemImage (HTREEITEM hItem , int &nImage , int &nSelectedImage)

Retrieves the image of an entry.

hItem	Handle of the item whose image is to be retrieved.
nImage	An integer that receives the index of the item's image within the tree view control's image list.
nSelectedImage	An integer that receives the index of the item's selected image within the tree view control's image list.

Returns Nonzero if successful; otherwise 0.

Remarks: Each item in a tree view control can have a pair of bitmapped images associated with it. The images appear on the left side of an item's label. One image is displayed when the item is selected, and the other is displayed when the item is not selected. For example, an item might display an open folder when it is selected and a closed folder when it is not selected.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItem | GetItemImageEx | SetItemImageEx | SetImageList | GetImageList | SetItemImage

CTreeListCtrl::GetItemImageEx

BOOL GetItemImageEx (HTREEITEM hItem , int *pImage , int nCol = 0)

Retrieves the image of an entry.

hItem	Handle of the item whose image is to be retrieved.
pImage	Pointer whose retrieves the image number, or -1 if no image is asserted to the item.
nCol	Is the column of the item.

Returns TRUE if the image number was detected or FALSE if an error occurs.

int GetItemImageEx (HTREEITEM hItem , int nCol = 0)

Retrieves the image of an entry.

hItem	Handle of the item whose image is to be retrieved.
nCol	Is the column of the item.

Returns the image number or -1 no image is asserted to the item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItem | SetItemImageEx | SetImageList | GetImageList | GetItemImage | SetItemImage

CTreeListCtrl::GetItemState

UINT GetItemState (HTREEITEM hItem , UINT nStateMask , int nCol = 0)

Retrieve the state of an item.

hItem	The handle of a tree view control item.	
nStateMask	Mask indicating which states are to be retrieved:	
	TVIS_BOLD	The item is bold.
	TVIS_CUT	The item is selected as part of a cut-and-paste operation.
	TVIS_DROPHILITED	The item is selected as a drag-and-drop target.
	TVIS_EXPANDED	The item's list of child items is currently expanded; that is, the child items are visible. This value applies only to parent items. This bits is only for the first column.
	TVIS_EXPANDEDONCE	The item's list of child items has been expanded at least once. The TVN_ITEMEXPANDING and TVN_ITEMEXPANDED notification messages are not generated for parent items that have this state set in response to a TVM_EXPAND message. Using TVE_COLLAPSE and TVE_COLLAPSERESET with TVM_EXPAND will cause this state to be reset. This value applies only to parent items. This bits is only for the first column.
	TVIS_EXPANDPARTIAL	A partially expanded tree list item. In this state, some, but not all, of the child items are visible and the parent item's plus symbol is displayed. This bits is only for the first column.
	TVIS_SELECTED	The item is selected. Its appearance depends on whether it has the focus. The item will be drawn using the system colors for selection. This bits is only for the first column.
	TVIS_OVERLAYMASK	Mask for the bits used to specify the item's overlay image index. I this bits are not zero, a overlay icon is drawn in front of the common icon.
	TVIS_STATEIMAGEMASK	Mask for the bits used to specify the item's state image index. This bits are used to get the check box state too.
	TVIS_USERMASK	Same as TVIS_STATEIMAGEMASK.
	TVIS_UNTERLINE	The item text is underlined.
nCol	Selects the column of the item entry.	

Returns the state bits of the selected item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItemState | SetItemCheckBox | GetItemCheckBox

CTreeListCtrl::GetItemText

BOOL GetItemText (HTREEITEM hItem , LPTSTR pBuffer , int iMax , int nCol = 0)
LPCTSTR GetItemText (HTREEITEM hItem , int nCol = 0)

Gets the text of an item and stores it in an buffer, or retrieve a pointer to the text.

hItem	Is the handle of the item.
pBuffer	Is the text buffer where the text will be saved.
iMax	Is the size of the text buffer in chars.
nCol	Is the column of the item .

Returns a pointer to the text or TRUE, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | SetItemState | SetItemText

CTreeListCtrl::GetItemTextColor

COLORREF GetItemTextColor (HTREEITEM hItem , int nCol) const

Retrieves the text color of an item.

hItem	Handle of the item whose text color is to be retrieved.
nCol	Selects the column of the item raw.

Returns the RGB value of the text color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColor | GetItemBkColor | SetColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::GetLastChild

HTREEITEM GetLastChild (HTREEITEM hItem)

Retrieves the last child item of a tree item;

hItem	Is the handle of the tree item.

Retrieves the handle of the last child, or NULL if not success;

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFirstVisibleItem | GetFocusItem | GetNextSelected | GetNextSelectedChild | GetNextVisibleItem | GetNextItem

CTreeListCtrl::GetNextItem

HTREEITEM GetNextItem (HTREEITEM hItem , UINT nCode = TVGN_NEXT)

Call this function to retrieve the tree view item that has the specified relationship, indicated by the *nCode* parameter, to *hItem*.

hItem	Is the handle of a tree item.
nCol	Is the column of the item .

TVGN_CARET	Retrieves the currently selected item.
TVGN_CARETSUB	Retrieves the currently selected column.
TVGN_CHILD	Retrieves the first child item.
TVGN_DROPHILITE	Retrieves the item that is the target of a drag-and-drop operation.
TVGN_DROPHILITESUB	Retrieves the column that is the target of a drag-and-drop operation.
TVGN_FIRSTVISIBLE	Retrieves the first visible item.
TVGN_FOCUS	Retrieves the item which has the focus. If the window not have the focus, the first selected item is retrieved.
TVGN_FOCUSSUB	Retrieves the column which has the focus. If the window not have the focus, the selected column is retrieved.
TVGN_LASTCHILD	Retrieves the last child item.
TVGN_LASTVISIBLE	Retrieves the last expanded item in the tree. This does not retrieve the last item visible in the tree-view window.
TVGN_NEXT	Retrieves the next sibling item.
TVGN_NEXTITEM	Retrieves the next item. This is at first the child, then the sibling or at last the first sibling of the parent item.
TVGN_NEXTSELCHILD	Retrieves the next selected child item (see **TVS_EX_MULTISELECT**).
TVGN_NEXTSELECTED	Retrieves the next selected item (see **TVS_EX_MULTISELECT**).
TVGN_NEXTVISIBLE	Retrieves the next visible item that follows the specified item.
TVGN_PARENT	Retrieves the parent of the specified item.
TVGN_PREVIOUS	Retrieves the previous sibling item.
TVGN_PREVIOUSVISIBLE	Retrieves the first visible item that precedes the specified item.
TVGN_ROOT	Retrieves the first child item of the root item.

The handle of the next item if successful; otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetDropHilightColumn | GetFirstSelected | GetFirstVisibleItem | GetFocusColumn | GetFocusItem | GetNextSelected | GetNextSelectedChild | GetNextVisibleItem | GetChildItem | GetNextSiblingItem | GetPrevSiblingItem | GetParentItem

CTreeListCtrl::GetNextSiblingItem

HTREEITEM GetNextSiblingItem(HTREEITEM hItem)

Gets the next item on the same level.

hItem	Is the handle of the item.

Retrieves the handle of the next sibling item if successful; otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetNextVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetFirstSelected | GetChildItem | GetNextItem | GetPrevSiblingItem | GetParentItem

CTreeListCtrl::GetNextSelected

HTREEITEM GetNextSelected (HTREEITEM hItem)

Retrieves the handle of the next selected item.

hItem	Is the handle of the item, where the search should be started. Use **TVI_ROOT** to get the first selected item.

Returns the item handle, or NULL if no other selected item is present.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusItem | GetItemOfRow | GetNextSelectedChild | GetRowCount | GetRowOfItem | SelectItem | GetFocusColumn | GetSelectionColumn | Select

CTreeListCtrl::GetNextSelectedChild

HTREEITEM GetNextSelectedChild (HTREEITEM hItem)

Retrieves the handle of the next selected child item.

hItem	Is the handle of the parent item, where the search should be started. Use **TVI_ROOT** to start at the root entry.

Returns the item handle, or NULL if no other selected item in the childs is present.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusItem | GetItemOfRow | GetNextSelected | GetRowCount | GetRowOfItem | SelectItem | Select

CTreeListCtrl::GetNextVisibleItem

HTREEITEM GetNextVisibleItem(HTREEITEM hItem)

Gets the next visible item.

hItem	Is the handle of a visible item.

Retrieves the handle of the next visible item, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetPrevVisibleItem

CTreeListCtrl::GetParentItem

HTREEITEM GetParentItem(HTREEITEM hItem)

Gets the parent item of an entry.

hItem	Is the handle of the entry.

Retrieves the handle of the parent item, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetNextVisibleItem | GetChildItem | GetNextItem | GetNextSiblingItem | GetPrevSiblingItem

CTreeListCtrl::GetPrevSiblingItem

HTREEITEM GetPrevSiblingItem(HTREEITEM hItem)

Gets the previous item on the same level.

hItem	Is the handle of the item.

Retrieves the handle of the previous sibling item if successful; otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetNextVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetFirstSelected | GetChildItem | GetNextItem | GetNextSiblingItem | GetParentItem

CTreeListCtrl::GetPrevVisibleItem

HTREEITEM GetPrevVisibleItem(HTREEITEM hItem)

Gets the previous visible item.

hItem	Is the handle of a visible item.

Retrieves the handle of the previous visible item, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetNextVisibleItem

CTreeListCtrl::GetRootItem

HTREEITEM GetRootItem( )

Retrieves the handle of the root item, otherwise NULL

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstVisibleItem | EnsureVisible | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | GetNextVisibleItem

CTreeListCtrl::GetRowCount

int GetRowCount ( ) const

Retrieves the count rows. This count represents all expanded items.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemOfRow | GetRowOfItem | GetCountPerPage

CTreeListCtrl::GetSelectionColumn

int GetSelectionColumn ( )

Retrieves the column number of the entry which contains the selected item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusColumn | GetFocusItem

CTreeListCtrl::GetStyle

DWORD GetStyle ( ) const

Retrieves the style flags of the controls.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | Extended Styles | GetExtendedStyle | SetExtendedStyle | SetStyle

CTreeListCtrl::IsItemVisible

INT IsItemVisible (HTREEITEM hItem , int nCol = -1)

Checks if an item is visible.

hItem	Is the handle of the item.
nCol	Is the column which should be testet. A -1 value means only check the row.

Returns:

-1 Unknown item.
0 Item is not expanded.
1 Item is expanded but not visible.
2 Item is expanded and partial visible.
3 Item is expanded and the column is partial visible.
4 Item is expanded and visible.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | HitTest | GetItemRect

CTreeListCtrl::SetColor

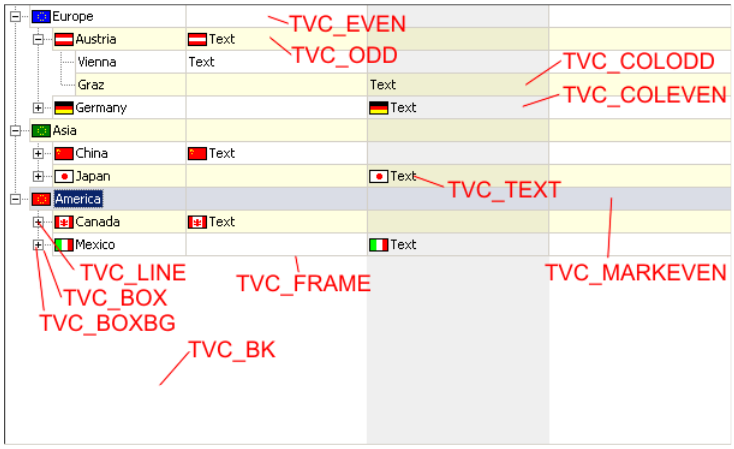
COLORREF SetColor (int iIndex , COLORREF uColor)

Changes the color of a specified display element.

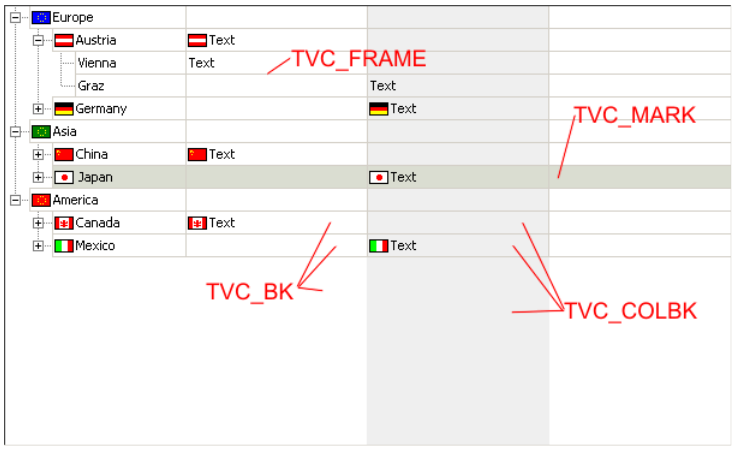
iIndex	Is the index of the display element: • **TVC_BK** background color. • **TVC_ODD** is the odd background color if TVS_EX_ALTERNATECOLOR is used. • **TVC_EVEN** is the even background color if TVS_EX_ALTERNATECOLOR is used. • **TVC_FRAME** is the color of the frame lines (with TVS_EX_ITEMLINES) • **TVC_TEXT** is the common text color • **TVC_LINE** is the color of the lines around the buttons • **TVC_BOX** is the inner color of the buttons (of the + - items) • **TVC_BOXBG** is the background color of the buttons • **TVC_TRACK** is the color of draged items • **TVC_MARK** is the color of a selected row • **TVC_MARK_ODD** is the color of a selected odd row • **TVC_MARK_EVEN** is the color of a selected even row • **TVC_INSERT** is the color of the insert mark
uColor	Is the RGB value of the new color. Use **TV_NOCOLOR** to set the default color.

Returns the specified color in the COLORREF format.

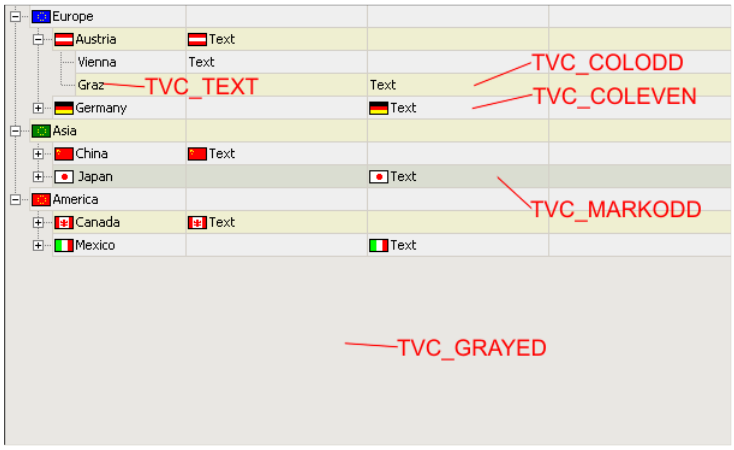
Colors if TVS_EX_ALTERNATECOLOR is set and column 2 is in the mark mode:



Colors if TVS_EX_ALTERNATECOLOR isn't set and column 2 is in the mark mode:



Colors if TVS_EX_ALTERNATECOLOR and TVS_EX_GRAYEDDISABLE is set, and the control is disabled:



see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColor | GetItemBkColor | GetItemTextColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::SetExtendedStyle

DWORD SetExtendedStyle (DWORD dwStyle)
DWORD SetExtendedStyle (DWORD dwStyle , DWORD dwMask)

Changes the extended style flags in the control.

dwStyle	Are the new extended style flags.
dwMask	This mask selects the extended style flags which should be changed.

Returns the new extended styles flags.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | GetExtendedStyle | GetStyle | SetStyle | Styles

CTreeListCtrl::SetFocusItem

BOOL SetFocusItem (HTREEITEM hItem , int nCol = -1)

Selects the item with the focus.

hItem	Handle of a tree item.
nCol	Is the column wich receives the focus. -1 means no change.

Returns TRUE if the focus was selected, FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusColumn | GetFocusItem | SelectItem

CTreeListCtrl::SetItemBkColor

COLORREF SetItemBkColor (HTREEITEM hItem , int nCol , COLORREF uColor)

Changes the background color of an item.

hItem	Is the item handle.
nCol	Is the column of the item. The value -1 means the full column.
uColor	Is the RGB value of the background color. Use TV_NOCOLOR to set the default background color of the control.

Returns the new color value of the item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColor | GetItemBkColor | GetItemTextColor | SetColor | SetItemTextColor

CTreeListCtrl::SetItemCheckBox

BOOL SetItemCheckBox (HTREEITEM hItem , int iState , int nCol = 0 , UINT uMask = 0x0F)

Sets the TVIS_STATEIMAGEMASK bits in an item, for the button state.

hItem	Is the handle of the Item.
iState	Is the new state of the button: 0 = not visible 1 = not selected 2 = selected If TVS_EX_BITCHECKBOX is enabeld: 0 = not selected 1 = selected
nCol	Is the colomn of the button.
uMask	Selects wich bits in the TVIS_STATEIMAGEMASK should be changed.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | SetItemState | GetItemCheckBox

CTreeListCtrl::SetItemState

BOOL SetItemState (HTREEITEM hItem , UINT nState , UINT nStateMask)
BOOL SetItemState (HTREEITEM hItem , int nCol , UINT nState , UINT nStateMask)

Sets the state bits of an Item

hItem	Is the handle of the item.
nCol	Is the column of the item.
nState	New state bits. Look at GetItemState for more details.
nStateMask	Mask of the bits which should be changed. Look at GetItemState for more details.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemText | SetItemBkColor | SetItemImageEx | SetItemText | SetItemTextColor | Expand | SetItemCheckBox | GetItemCheckBox

CTreeListCtrl::SetItemText

BOOL SetItemText (HTREEITEM hItem , LPCTSTR pText , int nCol = 0)

Sets the text of the item specified by *hItem*.

hItem	Handle of the item whose text is to be set.
pText	Address of a string containing the new text for the item.
nCol	Is the column for the text. (0=tree column)

Returns TRUE if the text was changed or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | GetItemText | SetItemBkColor | SetItemImageEx | SetItemState | SetItemTextColor

CTreeListCtrl::SetItemTextColor

COLORREF SetItemTextColor (HTREEITEM hItem , int nCol , COLORREF uColor)

Retrieves the background color of an item.

hItem	Handle of the item in the tree control with the item to set the color.
nCol	Is the column of the item. The value -1 means the full column.
uColor	Is the RGB value of the text color. Use **TV_NOCOLOR** to select the default text color.

Returns the RGB value of the text color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColor | GetItemBkColor | GetItemTextColor | SetColor | SetItemBkColor

CTreeListCtrl::SetStyle

DWORD SetStyle (DWORD dwStyle)
DWORD SetStyle (DWORD dwStyle , DWORD dwMask)

Call this member function to set the styles for a tree list control.

dwStyle	Are the new style flags for the control.

dwMask	This mask selects which style flags should be changed.

Returns the new style bits of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Create | Extended Styles | GetExtendedStyle | GetStyle | SetExtendedStyle

CTreeListCtrl::SortChildren

BOOL SortChildren (HTREEITEM hItem , BOOL bRecusive = FALSE)

Call this function to sort the child items of the given parent item in a tree view control.

hItem	Handle of the parent item whose child items are to be sorted. If *hItem* is **NULL**, sorting will proceed from the root of the tree.
bRecusive	If TRUE all childes of the childes will be sorted too. If false **SortChildren** will not recurse through the tree, only the immediate children of *hItem* will be sorted.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SortChildrenCB | SortChildrenEX

CTreeListCtrl::SortChildrenCB

BOOL SortChildrenCB (LPTVSORTCB pSort , BOOL bRecusive = FALSE)

Call this function to sort tree view items using an application-defined callback function that compares the items.

pSort	Pointer to a TVSORTCB structure.

```
typedef struct
    {
    HTREEITEM    hParent;
    PFNTVCOMPARE lpfnCompare;
    LPARAM       lParam;
    } TVSORTCB, *LPTVSORTCB;
```

Contains information used to sort child items in a tree view control. This structure is used with the **TVM_SORTCHILDRENCB** message. This structure is identical to the **TV_SORTCB** structure, but it has been renamed to follow current naming conventions.

hParent
Handle to the parent item.

lpfnCompare
Address of an application-defined callback function, which is called during a sort operation each time the relative order of two list items needs to be compared. The callback function has the following form:

```
int CALLBACK CompareFunc(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort);
```

The callback function must return a negative value if the first item should precede the second, a positive value if the first item should follow the second, or zero if the two items are equivalent.

The *lParam1* and *lParam2* parameters correspond to the **lParam** member of the TVITEM structure for the two items being compared. The *lParamSort* parameter corresponds to the **lParam** member of this structure.

lParam
Application-defined 32-bit value that gets passed as the *lParamSort* argument in the callback function specified in *lpfnCompare*.

bRecusive	If TRUE all childes of the childes will be sorted too. If false **SortChildrenCB** will not recurse through the tree, only the immediate children of *hItem* will be sorted.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SortChildren | SortChildrenEX

CTreeListCtrl::SortChildrenEX

BOOL SortChildrenEX (LPTVSORTEX pSort , BOOL bRecusive = FALSE)

Call this function to sort tree view items using an application-defined extended callback function that compares the items.

pSort — Pointer to a TVSORTEX structure.

```
typedef struct
    {
    HTREEITEM      hParent;
    PFNTVCOMPAREEX lpfnCompare;
    LPARAM         lParam;
    } TVSORTEX, *LPTVSORTEX;
```

Contains information used to sort child items in a tree view control. This structure is used with the **TVM_SORTCHILDRENEX** message. This structure is identical to the **TV_SORTEX** structure, but it has been renamed to follow current naming conventions.

hParent
Handle to the parent item.

lpfnCompare
Address of an application-defined callback function, which is called during a sort operation each time the relative order of two list items needs to be compared. The callback function has the following form:

```
int CALLBACK CompareFunc(HWND hWnd,HTREEITEM hItem1,HTREEITEM hItem2,LPARAM lParam1,LPARAM lParam2,LPARAM lParam);
```

The callback function must return a negative value if the first item should precede the second, a positive value if the first item should follow the second, or zero if the two items are equivalent.

The *lParam1* and *lParam2* parameters correspond to the **lParam** member of the TVITEM structure for the two items being compared. The *lParamSort* parameter corresponds to the **lParam** member of this structure. The hWnd member ist the window handle of the three list control, *hTree1* and *hTree2* are the tree list item which should be compared.

lParam
Application-defined 32-bit value that gets passed as the *lParamSort* argument in the callback function specified in *lpfnCompare*.

bRecusive — If TRUE all childes of the childes will be sorted too. If false **SortChildrenEX** will not recurse through the tree, only the immediate children of *hItem* will be sorted.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SortChildren | SortChildrenCB

CTreeListCtrl::CreateDragImage

CImageList *CreateDragImage (HTREEITEM hItem , int nCol = 0)

Creates a dragging bitmap for the specified tree view item.

hItem — Handle of the tree item to be dragged.

nCol — Is the column of the item.

Returns a pointer to the image list to which the dragging bitmap was added, if successful; otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SelectDropTarget | GetDropHilightColumn

CTreeListCtrl::DeleteAllItems

int DeleteDeleteAllItems ( )

Call this function to delete all items from the tree view control.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItem | InsertItem | DeleteColumn | DeleteItem | DeleteChildItems

CTreeListCtrl::DeleteChildItems

BOOL DeleteChildItems (HTREEITEM hItem)

Deletes all childs of an item.

hItem	Is the base item where the childs should be deleted.

Retuns TRUE if one ore more childs would be deleted.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteAllItems | DeleteColumn | DeleteItem

CTreeListCtrl::DeleteItem

int DeleteItem (HTREEITEM hItem)

Call this function to delete an item from the tree view control.

hItem	Handle of the tree item to be deleted. If *hitem* has the **TVI_ROOT** value, all items are deleted from the tree view control.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItem | InsertItem | DeleteColumn | DeleteAllItems | DeleteChildItems

CTreeListCtrl::DisableItemAutoEdit

BOOL DisableItemAutoEdit (HTREEITEM hItem , int nCol , BOOL bDisable = TRUE)

Disable or enable the auto-edit-option for an item.

hItem	Handle of the item whose auto-edit-option is to disabling.
nCol	Is the column of the item.
bDisable	If this parameter is TRUE, the option will be disabeld.

Returns nonzero if successful, otherwise zero.

The function sets the highest bit in the state-item-bits (means **TVAE_STATEBIT** = 0x8000). A high bit means that the auto-edit-option is disabled for the item, a low bit that the option is enabeld. If no **TVAE_STATEENABLE** bit is set for the column, the auto-edit-option is allways enabled.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumnAutoEdit | SetColumnAutoIcon | GetItemState | SetItemState

CTreeListCtrl::EditLabel

CEdit *EditLabel (HTREEITEM hItem , int nCol = 0 , int iFull = 0 , int iSel = 0)

Call this function to begin in-place editing of the specified item's text. The editing is accomplished by replacing the text of the item with a single-line edit control containing the text.

hItem	Handle of the tree item to be edited.
nCol	Is the column of the item to be edited.
iFull	Should the edit control be stretched over the full column.
iSel	Defines the text selection:

TVIR_SELAREA(from,to)	Selects a text area
TVIR_SETCURSOR(pos)	Sets the cursor to a position
TVIR_SETAT(pos)	Sets the cursor to a pixel position
0	Selects the full text

Returns if successful, a pointer to the CEdit object that is used to edit the item text, otherwise NULL.

If the text input was finished, a TVN_ENDLABELEDIT notify message will be send. With the result on this message, it can be decided if the new text will be transfered to the control.

```
ON_NOTIFY(TVN_ENDLABELEDIT,IDC_TREELIST,OnEndLabelEdit)

...

void CTreeListDlg::OnEndLabelEdit(NMHDR *pNmHdr,LRESULT *pResult)
{
NMTVDISPINFO *pHeader = (NMTVDISPINFO*)pNmHdr;
CString       sText;
HTREEITEM     hItem;
unsigned      uCol;


sText = pHeader->item.pszText;
hItem = pHeader->item.hItem;
uCol  = pHeader->item.cChildren;

if(pHeader->item.mask&TVIF_TEXTCHANGED)
     {
     // the text of the item was changed
     }

if(pHeader->item.mask&TVIF_RETURNEXIT)
     {
     // the user has pressed RETURN to input the text
     }

if(AccepptTheInput())
       *pResult = 0;   // confirm the new text
else   *pResult = 1;   // don't accept the new text
}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabelCb | SetColumnAutoEdit | StartEdit

CTreeListCtrl::EditLabelCb

CComboBox *EditLabelCb (HTREEITEM hItem , int nCol = 0 , int iFull = 0 , int iList = 0 , int iSel = 0)

Call this function to begin in-place editing of the specified item's text. The editing is accomplished by replacing the text of the item with a combo box control containing the text.

hItem	Handle of the tree item to be edited.
nCol	Is the column of the item to be edited.
iFull	Should the edit control be stretched over the full column.
iList	If this parameter isn't zero, the combo box, contains only a listbox.
iSel	Defines the text selection:

TVIR_SELAREA(from,to)	Selects a text area
TVIR_SETCURSOR(pos)	Sets the cursor to a position
TVIR_SETAT(pos)	Sets the cursor to a pixel position
0	Selects the full text

Returns if successful, a pointer to the CComboBox object that is used to edit the item text, otherwise NULL.

Use this pointer to fill the combo box with items.

If the text input was finished, a TVN_ENDLABELEDIT notify message will be send. With the result on this message, it can be decided if the new text will be transfered to the control.

```
ON_NOTIFY(TVN_ENDLABELEDIT,IDC_TREELIST,OnEndLabelEdit)

...

void CTreeListDlg::OnEndLabelEdit(NMHDR *pNmHdr,LRESULT *pResult)
{
NMTVDISPINFO *pHeader = (NMTVDISPINFO*)pNmHdr;
CString       sText;
HTREEITEM     hItem;
unsigned      uCol;


sText = pHeader->item.pszText;
hItem = pHeader->item.hItem;
uCol  = pHeader->item.cChildren;

if(pHeader->item.mask&TVIF_TEXTCHANGED)
     {
     // the text of the item was changed
     }

if(pHeader->item.mask&TVIF_RETURNEXIT)
     {
     // the user has pressed RETURN to input the text
     }

if(AccepptTheInput())
       *pResult = 0;   // confirm the new text
else   *pResult = 1;   // don't accept the new text
}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabel | SetColumnAutoEdit | StartEdit

CTreeListCtrl::EnsureVisible

BOOL EnsureVisible (HTREEITEM hItem)
BOOL EnsureVisible (HTREEITEM hItem , int nCol)

Call this function to ensure that a tree view item is visible. If necessary, the function expands the parent item or scrolls the tree view control so that the item is visible.

hItem	Handle of the tree item being made visible.
nCol	Column of the tree item being made visible.

Returns TRUE if the system scrolled the items in the tree-view control to ensure that the specified item is visible. Otherwise, the return value is FALSE

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemRect | SelectSetFirstVisible | GetFirstVisibleItem | GetNextVisibleItem

CTreeListCtrl::Expand

int Expand (HTREEITEM hItem , UINT nCode)

Call this function to expand or collapse the list of child items, if any, associated with the given parent item.

hItem	Handle of the tree item being expanded.

nCode	A flag indicating the type of action to be taken. This flag can have one of the following values: • **TVE_COLLAPSE** Collapses the list. • **TVE_COLLAPSERESET** Collapses the list and removes the child items. • **TVE_EXPAND** Expands the list. • **TVE_TOGGLE** Collapses the list if it is currently expanded or expands it if it is currently collapsed. Flags which can use with *TVE_EXPAND.* • **TVIS_EXPANDPARTIAL** Set *TVIS_EXPANDPARTIAL* at expanding • **TVE_EXPANDRECURSIVE** Expand or colapses all parents too • **TVE_EXPANDFORCE** Don't remove partial flag if expanded • **TVE_EXPANDNEXT** Begin at the next parent above • **TVE_ALLCHILDS** Repeat action on all childs • **TVE_ONLYCHILDS** Start with action at first child

Returns nonzero if successful, otherwise 0.

Use the flag *TVE_ONLYCHILDS* only with the flag *TVE_ALLCHILDS*. The flag *TVE_ALLCHILDS* can't be used with *TVE_EXPANDRECURSIVE*.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItem | GetItemState | SetItemState | ExpandAll | CollapseAll

CTreeListCtrl::ExpandAll

BOOL ExpandAll (HTREEITEM hItem , int iFlags = 0)

Expands all parent items of an entry.

hItem	Handle of the entry
iFlags	Several options • **TVIS_EXPANDPARTIAL** Set *TVIS_EXPANDPARTIAL* at expanding • **TVE_EXPANDFORCE** Set *TVIS_EXPANDPARTIAL* if not expanded at expanding

Returns TRUE if succefull

see also: CTreeListCtrl Overview | CTreeListCtrl Class Members | Expand | GetItemState | GetParentItem | CollapseAll

CTreeListCtrl::FindItem

HTREEITEM FindItem (HTREEITEM hItem , int iFlags , int nCol , LPCTSTR pText , LPARAM lParam , UINT uState , UINT uStateMask)

Seaches for an item in the control with several properties.

hItem	Is the item handle where the seach begins.
iFlags	This flag define the seach options: **TVIF_CHILD** begin at the first child of hItem to search **TVIF_NEXT** begin at the next item after hItem to search **TVIF_PARAM** compare the the lParam entry at the search **TVIF_TEXT** compare the the pText entry at the search **TVIF_STATE** compare the the uState entry at the search **TVIF_CASE** don't differ between lower and upper case
nCol	Selects the column for the text comparison.
pText	Is the text option for the seach.
lParam	Is the param option for the seach.

uState	Are the state bits for the seach. See at GetItemState for more informations.
uStateMask	Only bits which are set in this mask will be compared.

Returns the handle of the item which was found, or NULL if no item was found.

HTREEITEM FindItem (HTREEITEM hItem , LPCTSTR pText , int nCol = 0 , int iCase = 0)

Seaches for an item in the control with which a specified text.

hItem	Is the parent item handle where the seach begins.
pText	Is the text option for the seach.
nCol	Selects the column for the text comparison.
iCase	Differ between lower and upper case. (0=yes 1=no)

Returns the handle of the item which was found, or NULL if no item was found.

HTREEITEM FindItem (HTREEITEM hItem , LPARAM lParam)

Seaches for an item in the control with a specified user data value.

hItem	Is the item handle where the seach begins.
lParam	Is the param option for the seach.

Returns the handle of the item which was found, or NULL if no item was found.

HTREEITEM FindItem (HTREEITEM hItem , TV_FIND pFind)

Seaches for an item in the control with a specified user data value.

hItem	Is the item handle where the seach begins.
pFind	Is pointer to the search options:

```
typedef struct
    {
    UINT    uFlags;
    UINT    uColumn;
    UINT    uState;
    UINT    uStateMask;
    LPARAM  lParam;
    LPCTSTR pText;
    } TV_FIND;
```

uFlags
This flag define the seach options:

TVIF_CHILD	begin at the first child of hItem to seach
TVIF_NEXT	begin at the next item after hItem to seach
TVIF_PARAM	compare the the lParam entry at the seach
TVIF_TEXT	compare the the pText entry at the seach
TVIF_STATE	compare the the uState entry at the seach
TVIF_CASE	don't differ between lower and upper case

uColum
Selects the column for the text comparison.
uState
Are the state bits for the seach. See at GetItemState for more informations.
uStateMask
Only bits which are set in this mask will be comared.
lParam
Is the param option for the seach.
pText
Is the text option for the seach.

Returns the handle of the item which was found, or NULL if no item was found.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemText

CTreeListCtrl::GetItemRect

BOOL GetItemRect (HTREEITEM hItem , RECT *pRect , int iCode)
BOOL GetItemRect (HTREEITEM hItem , int nCol , RECT *pRect , int iCode)

Call this function to retrieve the bounding rectangle for *hItem* and determine whether it is visible or not.

hItem	The handle of a tree view control item.
nCol	Selects the column from which the rectangle should be retrieved. If **TVIR_GETCOLUMN** in *iCode* isn't set, the full row is used to get the rectangle.
pRect	Pointer to a *RECT* structure that receives the bounding rectangle. The coordinates are relative to the upper-left corner of the tree view control.
iCode	This flags define which part of the item should be retrieved. **TVIR_TEXT** : retrieve only the text rectangle **TVIR_GETCOLUMN** : retrieve only the column rectangle

Returns nonzero if the item is visible, with the bounding rectangle contained in *pRect*. Otherwise, 0 with *pRect* uninitialized.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EnsureVisible | HitTest

CTreeListCtrl::HitTest

HTREEITEM HitTest (CPoint sPoint , UINT *pFlags)
HTREEITEM HitTest (TVHITTESTINFO *pHitTestInfo)

Call this function to determine the location of the specified point relative to the client area of a tree view control.

When this function is called, the *pt* parameter specifies the coordinates of the point to test. The function returns the handle of the item at the specified point or **NULL** if no item occupies the point. In addition, the *pFlags* parameter contains a value that indicates the location of the specified point.

sPoint	Client coordinates of the point to test.
pFags	Pointer to an integer that receives information about the results of the hit test. It can be one or more of the values listed under the **flags** member in the Remarks section.
pHitTestInfo	Address of a *TVHITTESTINFO* structure that contains the position to hit test and that receives information about the results of the hit test.

```
typedef struct
    {
    POINT     pt;
    UINT      flags;
    HTREEITEM hItem;
    }TVHITTESTINFO, *LPTVHITTESTINFO;
```

Contains information used to determine the location of a point relative to a tree view control. This structure is used with the *TVM_HITTEST* message. The structure is identical to the *TV_HITTESTINFO* structure, but it has been renamed to follow current naming conventions.

pt

Client coordinates of the point to test.

flags

Variable that receives information about the results of a hit test. This member can be one or more of the following values:

TVHT_ABOVE	Above the client area.
TVHT_BELOW	Below the client area.
TVHT_NOWHERE	In the client area, but below the last item.
TVHT_ONITEM	On the bitmap or label associated with an item.
TVHT_ONITEMBUTTON	On the button associated with an item.
TVHT_ONITEMICON	On the bitmap associated with an item.
TVHT_ONITEMINDENT	In the indentation associated with an item.
TVHT_ONITEMLABEL	On the label (string) associated with an item.
TVHT_ONITEMRIGHT	In the area to the right of an item.
TVHT_ONITEMSTATEICON	On the state icon for a tree view item that is in a user-defined state.
TVHT_ONSUBITEM	On a sub item icon or string.
TVHT_ONSUBICON	On a sub item icon.
TVHT_ONSUBLABEL	On a sub item string.
TVHT_ONSUBRIGHT	On a sub item after the text.
TVHT_ONRIGHTSPACE	On right space after columns.
TVHT_TOLEFT	To the left of the client area.
TVHT_TORIGHT	To the right of the client area.
Bit 24..31	Contains the column number. (use ***TVIR_COLTOSUB(..)***)

hItem
Handle to the item that occupies the point.

Returns the handle of the tree view item that occupies the specified point or **NULL** if no item occupies the point.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemRect | IsItemVisible

CTreeListCtrl::InsertItem

HTREEITEM InsertItem (LPTVINSERTSTRUCT pInsertStruct)

HTREEITEM InsertItem (UINT nMask , LPCTSTR pText , int nImage , int nSelectedImage, UINT nState , UINT nStateMask , LPARAM lParam, HTR
HTREEITEM InsertItem (UINT nMask , LPCTSTR pText , int nImage , int nSelectedImage, UINT nState , UINT nStateMask , LPARAM lParam, PFN

HTREEITEM InsertItem (LPCTSTR pText , int nImage , int nSelectedImage , HTREEITEM hParent = TVI_ROOT , HTREEITEM hInsertAfter = TVI_
HTREEITEM InsertItem (LPCTSTR pText , int nImage , HTREEITEM hParent = TVI_ROOT , HTREEITEM hInsertAfter = TV
HTREEITEM InsertItem (LPCTSTR pText , HTREEITEM hParent = TVI_ROOT , HTREEITEM hInsertAfter = TV

HTREEITEM InsertItem (LPCTSTR pText , int nImage , int nSelectedImage , PFNTVSORTEX pCmpProc , HTREEITEM hParent = TVI_ROOT , LPAR
HTREEITEM InsertItem (LPCTSTR pText , int nImage , PFNTVSORTEX pCmpProc , HTREEITEM hParent = TVI_ROOT , LPA
HTREEITEM InsertItem (LPCTSTR pText , PFNTVSORTEX pCmpProc , HTREEITEM hParent = TVI_ROOT , LP

Call this function to insert a new item in a tree view control.

pInsertStruct — A pointer to a TVINSERTSTRUCT that specifies the attributes of the tree view item to be inserted.

```
typedef struct
   {
   HTREEITEM  hParent;
   HTREEITEM  hInsertAfter;
   TVITEM     item;
   }TVINSERTSTRUCT,LPTVINSERTSTRUCT;
```

Contains information used to add a new item to a tree view control. This structure is used with the **TVM_INSERTITEM** message. The structure to the **TV_INSERTSTRUCT** structure, but it has been renamed to follow current naming conventions.

hParent
Handle to the parent item. If this member is the **TVI_ROOT** value or NULL, the item is inserted at the root of the tree view control.
hInsertAfter
Handle to the item after which the new item is to be inserted, or one of the following values:

TVI_FIRST	Inserts the item at the beginning of the list.
TVI_LAST	Inserts the item at the end of the list.
TVI_SORT	Inserts the item into the list in alphabetical order.
TVI_SORTEX	Inserts the with an order which is produce by a callback function. The function is set on *item.hItem*
TVI_AFTER	Inserts the item behind the *hParent* item.
TVI_BEFORE	Inserts the item before the *hParent* item.
TVI_ROW(n)	Inserts the item into the specific row n below the parent.

item
TVITEM structure that contains information about the item to add.

nMask — Integer specifying which attributes to set. See at the TVITEM structure in the Platform SDK.

pText — Address of a string containing the item's text.

nImage — Index of the item's image in the tree view control's image list.

nSelectedImage — Index of the item's selected image in the tree view control's image list.

nState — Specifies values for the item's states. See at SetItemState for a list of appropriate states.

nStateMask — Specifies which states are to be set. See the TVITEM structure in the Platform SDK.

lParam — A application-specific value associated with the item.

hParent — Handle of the inserted item's parent.

hInsertAfter — Handle of the item after which the new item is to be inserted.

pCmpProc — Is the sort function which is used to find the insert position.

```
int CALLBACK CompareFunc(HWND hWnd,HTREEITEM hItem,LPCTSR pTextItem,LPCTSR pTextInsert,LPARAM lParamItem,LPARAM lPara
```

hWnd
Is the window handle of the control.

hItem
Is the handle of the item which is currently compared.

pTextItem

Is the pointer to the of the item.

pTextInsert
Is the pointer to the of the call (equal to *pText*).

lParamItem
Is parameter value of the item.

lParamInsert
Is parameter value of the function (equal to *lParam*).

Return values off the callback:
0 = equal
>0 = insert new item is after the the item
<0 = insert new item is before the the item

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteItem | SetItem

CTreeListCtrl::SetImageList

CImageList SetImageList (CImageList *pImageList , int nImageListType)
HIMAGELIST SetImageList (HIMAGELIST hImageList , int nImageListType)

Call this function to set the index of the item's image within the tree list view control's image list.

pImageList / hImageList	Pointer to the image list to assign. If pImageList is NULL, all images are removed from the tree view control.
nImageListType	Type of image list to set. The image list can be one of the following values:

TVSIL_NORMAL	Sets the normal image list, which contains the selected and nonselected images for the tree view item.
TVSIL_STATE	Sets the state image list, which contains the images for tree view items that are in a user-defined state.
TVSIL_CHECK	Sets the image list, which contains the ckeckboxes for the auto-column-edit-mode.
TVSIL_SUBIMAGES	Sets the image list, for the sub columns (2...n). If this list is zero the normal image list is used for the columns.If in the main column an image index above 0x40000000 is defined, this image list is also be used.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItemImageEx | GetImageList | GetItemImage | SetItemImage

CTreeListCtrl::SetItem

BOOL SetItem (TVITEM *pItem)

Call this function to set the attributes of the specified tree view item.

pItem — A pointer to a TVITEM structure that contains the new item attributes, as described in the Platform SDK.

```
typedef struct
    {
    UINT      mask;
    HTREEITEM hItem;
    UINT      state;
    UINT      stateMask;
    LPTSTR    pszText;
    int       cchTextMax;
    int       iImage;
    int       iSelectedImage;
    int       cChildren;
    LPARAM    lParam;
    }TVITEM,*LPTVITEM;
```

For more details look at CTreeCtrl::SetItem in the MFC documentaion.

One difference between the CTreeCtrl::SetItem functionality is the *cChildren* member.
If the flag TVIF_SUBITEM in *mask* is set, the *cChildren* member contains the column number.

Returns TRUE if the text was changed or FALSE if an error occurs.

BOOL SetItem (HTREEITEM hItem , int nCol , UINT nMask , LPCTSTR pText , int nImage , int nSelectedImage , UINT nState , UINT nStateMask ,

Changes some properties of an item.

hItem	Is the handle of the item
nCol	Is the column for the text (0=tree column)
nMask	Defies which properties should be set (**TVIF_????**)
pText	Is the new text of the item (add **TVIF_TEXT** to nMask)
nImage	Is the new image for the item (add **TVIF_IMAGE** to nMask)
nSelectedImage	Is the new image for the item (add **TVIF_SELECTEDIMAGE** to nMask)
nState	Is the new state for the item (add **TVIF_STATE** to nMask)
nStateMask	Is a mask for the states
lParam	Is the new LPARAM parameter for the item (add **TVIF_PARAM** to nMask)

Returns TRUE if the text was changed or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | GetItemText | SetItemBkColor | SetItemImageEx | SetItemState | SetItemText | SetItemTextColor

CTreeListCtrl::SetItemImage

BOOL SetItemImage (HTREEITEM hItem , int nImage , int nSelectedImage)

Call this function to set the index of the item's image and its selected image within the tree view control's image list.

hItem	Handle of the item whose image is to be set.
nImage	Index of the item's image in the tree view control's image list.
nSelectedImage	Index of the item's selected image in the tree view control's image list.

Returns nonzero if successful, otherwise 0.

Remarks: Each item in a tree view control can have a pair of bitmapped images associated with it. The images appear on the left side of an item's label. One image is displayed when the item is selected, and the other is displayed when the item is not selected. For example, an item might display an open folder when it is selected and a closed folder when it is not selected.

If in the main column an image index above 0x40000000 is defined, for such values the sub image list is used with the index (n-0x40000000).

For more information on images, see CImageList.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | GetItemImage | SetItemImageEx

CTreeListCtrl::SetItemImageEx

BOOL SetItemImageEx (HTREEITEM hItem , int nImage , int nCol)

Call this function to set the index of the item's image within the tree list view control's image list.

hItem	Handle of the item whose image is to be set.
nImage	Index of the item's image in the tree view control's image list.
nCol	Index of the item's selected image in the tree view control's image list.

Returns nonzero if successful, otherwise 0.

Remarks: If in the main column an image index above 0x40000000 is defined, for such values the sub image list is used with the index (n-0x40000000).

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItem | SetImageList | GetImageList | GetItemImage | SetItemImage

CTreeListCtrl::Select

int Select (HTREEITEM hItem , UINT nCode)

Call this function to select the given tree view item, scroll the item into view, or redraw the item in the style used to indicate the target of a drag-and-drop operation.

If *nCode* contains the value **TVGN_CARET**, the parent window receives the **TVN_SELCHANGING** and **TVN_SELCHANGED** notification messages. In addition, if the specified item is the child of a collapsed parent item, the parent's list of child items is expanded to reveal the specified item. In this case, the parent window receives the **TVN_ITEMEXPANDING** and **TVN_ITEMEXPANDED** notification messages.

hItem	Handle of a tree item.
nCode	The type of action to take. This parameter can be one of the following values: • **TVGN_CARET** Sets the selection to the given item. • **TVGN_DROPHILITE** Redraws the given item in the style used to indicate the target of a drag-and-drop operation. • **TVGN_FIRSTVISIBLE** Scrolls the tree view vertically so that the given item is the first visible item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetItem | GetItemState | SetItemState | GetFirstSelected | GetNextSelected | GetNextSelectedChild | SelectChilds

CTreeListCtrl::SelectChilds

BOOL SelectChilds (HTREEITEM hItem = TVI_ROOT , int iMode = TVIS_WITHCHILDS)

This function selects or deselects all childs of an item.

hItem	Is the handle of the base item.
iMode	Is the mode for the action: **TVIS_WITHCHILDS** Selects the lower child too **TVIS_DESELECT** Deselects the items

Returns TRUE if succesfull or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Select | SelectDropTarget | SelectItem | SelectSetFirstVisible

CTreeListCtrl::SelectDropTarget

BOOL SelectDropTarget (HTREEITEM hItem , int nCol = 0)

Call this function to redraw the item in the style used to indicate the target of a drag-and-drop operation.

hItem	Handle of a tree item.
nCol	Column of the thee item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | CreateDragImage | GetDropHilightColumn

CTreeListCtrl::SelectItem

BOOL SelectItem (HTREEITEM hItem , int nCol = 0)

Call this function to select the given tree view item. If *hItem* is **NULL**, then this function selects no item.

hItem	Handle of a tree item.
nCol	Is the item column.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusItem | GetItemOfRow | GetNextSelected | GetNextSelectedChild | GetRowCount | GetRowOfItem | SelectChilds

CTreeListCtrl::SelectSetFirstVisible

int SelectSetFirstVisible (HTREEITEM hItem)

Call this function to scroll the tree view vertically so that the given item is the first visible item. The function sends a message to the window with the **TVM_SELECTITEM** and **TVGN_FIRSTVISIBLE** message parameters.

hItem	Handle of the tree item to be set as the first visible item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EnsureVisible

CTreeListCtrl::StartEdit

BOOL StartEdit (HTREEITEM hItem , int nCol = 0)

Starts a label edit via the TVN_STARTEDIT notify message.
You must handle this message in the parent window.

hItem	Handle of the tree item to be edited.
nCol	Is the column of the item to be edited.

Returns TRUE if the item will be edit.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabel | EditLabelCb | SetColumnAutoEdit | TVN_STARTEDIT

CTreeListCtrl::DeleteColumn

BOOL DeleteColumn (int nCol)

Deletes a colum from the header control.

nCol	Is the number of the column which should be deleted.

Returns nonzero if it was successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColumn | GetColumnCount | GetColumnWidth | SetColumn | SetColumnAutoEdit | SetColumnWidth

CTreeListCtrl::FixColumnSize

BOOL FixColumnSize (int nCol , BOOL bOn = TRUE , int iWidth = -1)

Fixes the size of a a column. The size of a fixed column can't be changed.

nCol	is the number of the column
bOn	is the fixed state
iWidth	sis the new width for the column. -1 means don't change width. TVCF_LASTSIZE restore size befor fixing.

Returns TRUE if succesfull or FALSE if an error occurs.

```
cMxControl.FixColumnSize(2,TRUE ,0            );   // hides the column 2
cMxControl.FixColumnSize(2,FALSE,TVCF_LASTSIZE);   // restores the column
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColumnWidth | SetColumnWidth

CTreeListCtrl::GetColumn

BOOL GetColumn (int nCol , TV_COLUMN *pColumn)

Retrieves the attributes of a tree list view control's column.

nCol	Index of the column whose attributes are to be retrieved.
pColumn	Address of a **TV_COLUMN** structure that contains the new column attributes, as described in the Platform SDK. The structure's mask member specifies which column attributes to set. If the mask member specifies the **TVCF_TEXT** value, the structure's *pszText* member is the address of a null-terminated string and the structure's *cchTextMax* member is ignored.

```
typedef struct
    {
    UINT   mask;
    int    fmt;
    int    cx;
    LPTSTR pszText;
    int    cchTextMax;
    int    iSubItem;
    int    iImage;
    int    iOrder;
    } TV_COLUMN;
```

mask
Variable specifying which members contain valid information. This member can be zero, or one or more of the following values:

TVCF_FMT	The **fmt** member is valid.
TVCF_IMAGE	The **iImage** member is valid.
TVCF_TEXT	The **pszText** member is valid.
TVCF_WIDTH	The **cx** member is valid. (-1=auto width)

fmt
Alignment of the column heading and the subitem text in the column. This member can be one of the following values:

TVCFMT_BITMAP_ON_RIGHT	The bitmap appears to the right of text. This does not affect an image from an image list assigned to the header item.
TVCFMT_CENTER	Text is centered.
TVCFMT_COL_HAS_IMAGES	The header item contains an image in the image list.
TVCFMT_IMAGE	The item displays an image from an image list.
TVCFMT_LEFT	Text is left-aligned.
TVCFMT_RIGHT	Text is right-aligned.

cx
Width of the column, in pixels.
pszText
If column information is being set, this member is the address of a null-terminated string that contains the column heading text. If the structure is receiving information about a column, this member specifies the address of the buffer that receives the column heading text.
cchTextMax
Size of the buffer pointed to by the **pszText** member. If the structure is not receiving information about a column, this member is ignored.
iSubItem
Is an used parameter.
iImage
Zero-based index of an image within the image list. The specified image will appear within the column.
iOrder
Is an used parameter.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumnCount | GetColumnWidth | SetColumn | SetColumnAutoEdit | SetColumnWidth | GetHeaderCtrl | InsertColumn

CTreeListCtrl::GetColumnCount

int GetColumnCount ( ) const

Returns the count of columns in the tree list view.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnWidth | SetColumn | SetColumnAutoEdit | SetColumnWidth

CTreeListCtrl::GetColumnOrderArray

BOOL GetColumnOrderArray (int iCount , int *pArray)

Gets the left-to-right order of columns from a tree list view control.

iCount	The number of columns in the tree list control.
pArray	A pointer to an array which receives the order array. For example, if the contents of the array are {0,2,1}, the control displays column 0, column 2, and column 1, from left to right. The first entry must be zero.

Returns TRUE if done or FALSE if failes.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumnOrderArray

CTreeListCtrl::GetColumnWidth

int GetColumnWidth (int nCol)

Get the width of a column.

nCol	Is the number of the column.

Returns the width of the column in pixels, or 0 if not succesfull.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnCount | SetColumn | SetColumnAutoEdit | SetColumnWidth

CTreeListCtrl::GetHeaderCtrl

CHeaderCtrl *GetHeaderCtrl ( ) const
VOID GetHeaderCtrl (HWND &hWnd) const

Returns a pointer or the window handle to the header control, used by the tree list control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColumn | GetColumnCount | GetColumnWidth | SetColumn | SetColumnAutoEdit | SetColumnWidth

CTreeListCtrl::InsertColumn

int InsertColumn (int nCol , const TV_COLUMN *pColumn)

Inserts a column in the header of the Tree-List-Control.

nCol	Position where the column should be inserted. Use -1 to at the column at end of the column bar.

pColumn	Is a pointer **TV_COLUMN** structure which contains the infos for the new column.
	
	`typedef struct`
	`{`
	`UINT mask;`
	`int fmt;`
	`int cx;`
	`LPTSTR pszText;`
	`int cchTextMax;`
	`int iSubItem;`
	`int iImage;`
	`int iOrder;`
	`}TV_COLUMN;`
	
	For more details look in the MFC documentation at CListViewCtrl::SetItem.
	
	Use the **TVCF_VWIDTH** macro in *mask* to define an auto expanding column.
	Use the **TVCF_MIN** macro in *mask* to get the minimal size from the *iOrder* member.
	Use the **TVCF_MARK** macro in *mask* to enable the **TVCFMT_MARK** bit in the *fmt* member.
	
	A negative value of the minimal size means that *abs(nMin)* represents the
	minimal size only for dynamic size changing.

Returns the position of the column or -1 if an error occurs.

int InsertColumn (int nCol , LPCTSTR pColumnText , int nFormat = TVCFMT_LEFT, int nWidth = -1 , int nMin = 0x8000)

Inserts a column in the header of the Tree-List-Control.

nCol	Position where the column should be inserted. Use -1 to at the column at end of the column bar.
pColumnText	Displayed text for the column.
nFormat	Is the format for the text • **TVCFMT_CENTER** centers the text in the column. • **TVCFMT_IMAGE** the column bar contains an image. • **TVCFMT_RIGHT** sets the column to right alignment. • **TVCFMT_LEFT** sets the column to left alignment. • **TVCFMT_MARK** sets mark mode for the column.
nWidth	Is the width for the column in pixels. A negative value means that the column is an auto expanding column. The value of *abs(nWidth)* represents the weight of the column. i.e.: *nWidth* for Col1 is -20 *nWidth* for Col2 is -30 *nWidth* for Col3 is -50 Control width is 500 pixels so Col1 gets 100 pixel (20% of 500) Col2 gets 150 pixel (30% of 500) Col3 gets 250 pixel (50% of 500)
nMin	Is the minimal size for the column. A negative value means that *abs(nMin)* represents the minimal size only for dynamic size changing. The user can set lower sizes in by mouse input this case. Use 0x8000 for no minimal size definition. i.e.: *nMin* is 50 So the user can only set a minimal width of 50 pixels By resizing the control the column gets a minimal width of 50 pixels *nMin* is -50 So the user can only set a minimal width of 0 pixels By resizing the control the column gets a minimal width of 50 pixels

Returns the position of the column or -1 if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColumn | GetHeaderCtrl | SetColumn | SetColumnAutoEdit

CTreeListCtrl::SetColumn

BOOL SetColumn (int nCol , const TV_COLUMN *pColumn)

Sets the attributes of a tree list view column.

nCol — Index of the column whose attributes are to be set.

pColumn — Address of a **TV_COLUMN** structure that contains the new column attributes, as described in the Platform SDK. The structure's mask member specifies which column attributes to set. If the mask member specifies the **TVCF_TEXT** value, the structure's *pszText* member is the address of a null-terminated string and the structure's *cchTextMax* member is ignored.

```
typedef struct
    {
    UINT   mask;
    int    fmt;
    int    cx;
    LPTSTR pszText;
    int    cchTextMax;
    int    iSubItem;
    int    iImage;
    int    iOrder;
    } TV_COLUMN;
```

mask
Variable specifying which members contain valid information. This member can be zero, or one or more of the following values:

TVCF_FMT	The **fmt** member is valid.
TVCF_IMAGE	The **iImage** member is valid.
TVCF_TEXT	The **pszText** member is valid.
TVCF_WIDTH	The **cx** member is valid. (-1=auto width)
TVCF_MARK	The **TVCFMT_MARK** bit in **fmt** is valid.

fmt
Alignment of the column heading and the subitem text in the column. This member can be one of the following values:

TVCFMT_BITMAP_ON_RIGHT	The bitmap appears to the right of text. This does not affect an image from an image list assigned to the header item.
TVCFMT_CENTER	Text is centered.
TVCFMT_COL_HAS_IMAGES	The header item contains an image in the image list.
TVCFMT_IMAGE	The item displays an image from an image list.
TVCFMT_LEFT	Text is left-aligned.
TVCFMT_RIGHT	Text is right-aligned.
TVCFMT_MARK	The column will be marked (darker color).

cx
Width of the column, in pixels.
pszText
If column information is being set, this member is the address of a null-terminated string that contains the column heading text. If the structure is receiving information about a column, this member specifies the address of the buffer that receives the column heading text.
cchTextMax
Size of the buffer pointed to by the **pszText** member. If the structure is not receiving information about a column, this member is ignored.
iSubItem
Is an used parameter.
iImage
Zero-based index of an image within the image list. The specified image will appear within the column.
iOrder
Is an unused parameter.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnCount | GetColumnWidth | SetColumnAutoEdit | SetColumnWidth | GetHeaderCtrl | InsertColumn | SetColumnImage | SetColumnText | SetColumnMark

CTreeListCtrl::SetColumnAutoEdit

int SetColumnAutoEdit (int nCol , int iMode = TVAE_EDIT)
int SetColumnAutoEdit (int nCol , int iMode , LPTSTR *pList , int iMax = 0)
int SetColumnAutoEdit (int nCol , int iMode , LPCTSTR *pList , int iMax = 0)
int SetColumnAutoEdit (int nCol , int iMode , LPCTSTR pText , int iMax = 0)
int SetColumnAutoEdit (int nCol , int iMode , TCHAR cChar , LPCTSTR pText , int iMax = 0)

Set the auto edit option for a column. With the auto edit option the control activates automatically an edit control or a combobox, if a user input occurs. An external notify message must not be parsed, and no EditLabel function must be called. **Note**: the style **TVS_EDITLABELS** must be enabled in the control.

nCol	Is the column for which the auto edit option should be set.
iMode	Here the auto edit option is configured:

TVAE_NONE	Disables auto edit option.
TVAE_EDIT	Enables the auto edit option with an edit control.
TVAE_COMBO	Enables the auto edit option with an editable combobox.
TVAE_CBLIST	Enables the auto edit option with a combobox without edit option.
TVAE_STEP	Enables the auto edit option with value step over an enter key input. If the text was changed a **TVN_STEPSTATECHANGED** notify message will be send.
TVAE_STEPED	Enables the auto edit option with value step over an ctlr+enter key input. If the text was changed a **TVN_STEPSTATECHANGED** notify message will be send. The user can change the text over an edit control via an enter key input.
TVAE_CHECK	Enables the auto edit option with checkboxes. The state of the checkbox can be get over the state-image-mask. (see GetItemState) The images for the checkbox could be set with *SetImageList(...,TVSIL_CHECK)* It is usefull to set the **TVAE_ICONCLICK** and **TVAE_DBLCLICK** too if checkboxes are used. If the text was changed a **TVN_CBSTATECHANGED** notify message will be send.
TVAE_CHECKED	Enables the auto edit option with checkboxes. The state of the checkbox can be get over the state-image-mask. (see GetItemState) The images for the checkbox could be set with *SetImageList(...,TVSIL_CHECK)* It is usefull to set the **TVAE_ICONCLICK** and and **TVAE_DBLCLICK** too if checkboxes are used. The user can change the text over an edit control via an enter key input. If the text was changed a **TVN_CBSTATECHANGED** notify message will be send.

This are extra options:

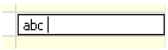
TVAE_DBLCLICK	Allows to start the edit function with a mouse double click.
TVAE_ONLYRETURN	Only the return key, starts the edit.
TVAE_NEXTLINE	Sets the selection mark into the next line if the input was finished with the enter key.
TVAE_FULLWIDTH	This option stretches the edit or combobox control to the full column width.
TVAE_STATEENABLE	If this option is enabled, the auto-edit can be disabled with the state bit **TVIS_DISABLEBIT** (0x8000) separate for each item. (use DisableItemAutoEdit)
TVAE_ICONCLICK	This means that a click on the item icon in the column, activates the autoedit. This is useful to simulate check boxes with the **TVAE_STEP** mode.
TVAE_DROPDOWN	If this option is enabled, the DropDownList of a ComboBox is automatically shown.

cChar	Is the used character to separate the entries in a text.
pText	Is a text which contains the entries for the listbox of the combo box, or the value stepper entries. The entries are separated with the character which is defined with *cChar*. If *cChar* isn't declared, the zero character will be used.
pList	Is a pointer list contains the entries for the listbox of the combo box, or the value stepper entries. Is *iMax* is set to zero, the end of the list is marked with a NULL pointer.
iMax	Is the count of entries in the *pList* or *pText* parameter. The zero value means that the count will be automatically detected.

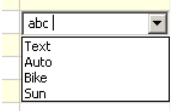
Returns TRUE if the auto edit option was set, otherwise FALSE.

i.e.:

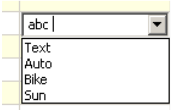
```
sMyControl.SetColumnAutoEdit(2,TVAE_EDIT|TVAE_FULLWIDTH);
```



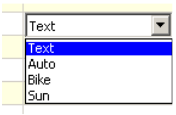

```
LPTSTR  *aList = {"Text","Auto","Bike","Sun",NULL};
sMyControl.SetColumnAutoEdit(2,TVAE_COMBOBOX|TVAE_FULLWIDTH,aList);
```




```
LPTSTR  *aList = {"Text","Auto","Bike","Sun","Dummy5","Dummy6"};
sMyControl.SetColumnAutoEdit(2,TVAE_COMBOBOX|TVAE_FULLWIDTH,aList,4);
```



```
LPTSTR  cText = "Text|Auto|Bike|Sun";
sMyControl.SetColumnAutoEdit(2,TVAE_CBLIST|TVAE_FULLWIDTH,'|',cText);
```




```
sMyControl.SetColumnAutoEdit(2,TVAE_CHECK|TVAE_ICONCLICK);
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnCount | GetColumnWidth | SetColumn | SetColumnWidth | EditLabelCb | SetColumnAutoIcon | DisableItemAutoEdit

CTreeListCtrl::SetColumnAutoIcon

int SetColumnAutoIcon (int nCol , int iIconOffset)

If the auto edit option for a column is enabled, this function sets the offset in the image list of the images which should be displayed. The resulting image is calculated by adding the offset and the selected entry in the combobox. If a string list is defined with SetColumnAutoEdit the offset is the index of the item text in the list.

nCol	Is the column for which the auto edit option should be set.
iIconOffset	Is the offset in the image list. -1 means no auto icon option.

Returns TRUE if the auto icon option was set, otherwise FALSE.

Note: If a text from the user input wasn't found in the defined string list, the icon (iIconOffset-1) will be set in the item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnCount | SetColumnAutoEdit

CTreeListCtrl::SetColumnImage

BOOL SetColumnImage (int nCol , int iImage)

Sets the image in the column header.

nCol	is the number of the column.
iImage	is the new image number.

Returns TRUE if succesfull.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumn | SetColumnWidth | SetColumnText

CTreeListCtrl::SetColumnMark

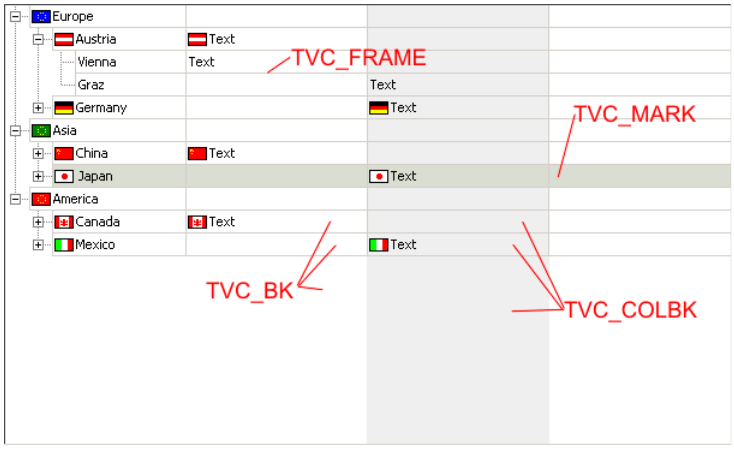
BOOL SetColumnMark (int nCol , BOOL bOn = TRUE)

Sets the mark state of a column. A marked column will be drawn darker.

nCol	is the number of the column.
bOn	is the mark state.

Returns TRUE if succesfull.

The next picture shows a marked column 2:



see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | InsertColumn | SetColumn

CTreeListCtrl::SetColumnOrderArray

BOOL SetColumnOrderArray (int iCount , const int *pArray)

Sets the left-to-right order of columns in a tree list view control. The use can change the displayed order of the sub columns. The first main column can't be moved.

iCount	The number of columns in the tree list control.
pArray	A pointer to an array specifying the order in which columns should be displayed, from left to right. For example, if the contents of the array are {0,2,1}, the control displays column 0, column 2, and column 1, from left to right. The first entry must be zero. If this pointer is NULL, the default order array will be set.

Returns TRUE if done or FALSE if failes.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetColumnOrderArray

CTreeListCtrl::SetColumnText

BOOL SetColumnText (int nCol , LPCTSTR pText)

Sets the text of an item in the column header

nCol	is the number of the column.
pText	is the new text for the column.

Returns TRUE if succesfull.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumn | SetColumnWidth | SetColumnImage

CTreeListCtrl::SetColumnWidth

BOOL SetColumnWidth (int nCol , int iWidth)

Changes the width of a column in tree list view.

nCol	Index of the column whose width is to be set.
iWidth	The new width of the column.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | DeleteColumn | GetColumn | GetColumnCount | GetColumnWidth | SetColumn | SetColumnAutoEdit | FixColumnSize

CTreeListCtrl::GetUserData

LPVOID GetUserData (HTREEITEM hItem) const

Retrieves the pointer to the user data field of a item. The user data field is an area which is added automatically to each item. The field has the same size for each item. The size of the field could be defined with the SetUserDataSize function.

hItem	Handle of the item whose user data pointer is to be retrieved.

Returns a pointer to the user data, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetUserDataSize

CTreeListCtrl::GetUserDataSize

INT GetUserDataSize ( ) const

Retrieves the size of the user data fields.

The user data field is an area which is added automatically to each item. The field has the same size for each item. The size of the field could be defined with the SetUserDataSize function.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetUserData

CTreeListCtrl::SetUserDataSize

INT SetUserDataSize (int iSize)

Changes the size of the user data field of the tree list items. The user data field is an area which is added automatically to each item. The field has the same size for each item. The size of the field could only be changed if the control is empty.

iSize	Is the new size of the user data field.

Returns the new size of the user data field, or 0 if an error occurs.

i.e.:

```
typedef struct
    {
    int  iValue;
    int  iState;
    char cText[32];
    }MyUserData;

MyUserData    *pData;
CTreeListCtrl *pTree;
HTREEITEM      hItem;
...

// Create control and set user data size
pTree->Create(TVS_HASLINES,CRect(0,0,0,0),this,1234);
pTree->SetUserDataSize(sizeof(MyUserData));

// Create an item and set user data
hItem =              pTree->InsertItem("TestEntry");
pData = (MyUserData*)pTree->GetUserData(hItem);
pData->iValue   = 5555;
pData->iState   = 6666;
pData->cText[0] = 'X';
pData->cText[1] =  0 ;
...

// Get the user data from an item
pData = (MyUserData*)pTree->GetUserData(hItem);
this->MessageBox(pData->cText);
...
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetUserData | GetUserDataSize

CTreeListCtrl::GetItemOfRow

HTREEITEM GetItemOfRow (int iRow) const

Retrieves the item handle of the item in a displayed row of the control.

iRow	Is the index of the raw.

Returns the item handle for the row, otherwise NULL.

This function is used in tree list controls, which are used as list control, to convert row indexes to item handles.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetRowCount | GetRowOfItem

CTreeListCtrl::GetRowOfItem

int GetRowOfItem (HTREEITEM hItem) const

Retrieves the displayed row from an item handle in the control.

hItem	Is the handle of the item.

Returns the index of the row of an item, otherwise -1 if the item is not expanded by his parent or not exist.

This function is used in tree list controls, which are used as list control, to convert item handles to row indexes.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemOfRow | GetRowCount | GetCountPerPage

CTreeListCtrl::ListCreateDragImage

CImageList *ListCreateDragImage (int iRow , int nCol = 0)

Creates a dragging bitmap for the specified tree view item.

iRow	Index of the row in the tree control with the item to be dragged.
nCol	Is the column of the item.

Returns a pointer to the image list to which the dragging bitmap was added, if successful; otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetDropHilightColumn | ListSelectDropTarget | SelectDropTarget

CTreeListCtrl::ListEditLabel

CEdit *ListEditLabel (int iRow , int nCol = 0 , int iFull = 0 , int iSel = 0)

Call this function to begin in-place editing of the specified item's text. The editing is accomplished by replacing the text of the item with a single-line edit control containing the text.

iRow	Index of the row in the tree control with the item to be edited.
nCol	Is the column of the item to be edited.
iFull	Should the edit control be stretched over the full column.
iSel	Defines the text selection:

TVIR_SELAREA(from,to)	Selects a text area
TVIR_SETCURSOR(pos)	Sets the cursor to a position
TVIR_SETAT(pos)	Sets the cursor to a pixel position
0	Selects the full text

Returns if successful, a pointer to the CEdit object that is used to edit the item text, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabel | EditLabelCb | ListEditLabelCb | SetColumnAutoEdit

CTreeListCtrl::ListDeleteItem

BOOL ListDeleteItem (int iRow)

Deletes an sItem in a Tree-List-Control which is used as List-Control

iRow	Is the row of the item in the tree list control.

Returns TRUE if the item was deleted or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | ListSetItem | ListInsertItem

CTreeListCtrl::ListEditLabelCb

CComboBox *ListEditLabelCb (int iRow , int nCol = 0 , int iFull = 0 , int iList = 0 , int iSel = 0)

Call this function to begin in-place editing of the specified item's text. The editing is accomplished by replacing the text of the item with a combo box control containing the text.

iRow	Index of the row in the tree control with the item to be edited.
nCol	Is the column of the item to be edited.
iFull	Should the edit control be stretched over the full column.
iList	If this parameter isn't zero, the combo box, contains only a listbox.
iSel	Defines the text selection:

TVIR_SELAREA(from,to)	Selects a text area
TVIR_SETCURSOR(pos)	Sets the cursor to a position
TVIR_SETAT(pos)	Sets the cursor to a pixel position
0	Selects the full text

Returns if successful, a pointer to the CComboBox object that is used to edit the item text, otherwise NULL.

Use this pointer to fill the combo box with items.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabel | EditLabelCb | ListEditLabel | SetColumnAutoEdit

CTreeListCtrl::ListEnsureVisible

BOOL ListEnsureVisible (int iRow)
BOOL ListEnsureVisible (int iRow , int nCol)

Call this function to ensure that a tree view item is visible. If necessary, the function expands the parent item or scrolls the tree view control so that the item is visible.

iRow	Index of the row in the tree control with the item being made visible.
nCol	Column of the tree item being made visible.

Returns TRUE if the system scrolled the items in the tree-view control to ensure that the specified item is visible. Otherwise, the return value is FALSE.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EnsureVisible | GetItemRect | ListGetItemRect

CTreeListCtrl::ListGetColor

BOOL ListGetColor (int iRow , int nCol , COLORREF &uBkColor , COLORREF &uTextColor)

Gets the colors of an item in a Tree-List-Control which is used as List-Control.

iRow	Is the row of the item.
nCol	Is the column of the item.
uBkColor	Is the new background color. Use **TV_NOCOLOR** to select the default color.
uTextColor	Is the new background color. Use **TV_NOCOLOR** to select the default color.

Returns TRUE if ok or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | ListGetItemBkColor | ListGetItemTextColor | ListSetColor | ListSetItemBkColor | ListSetItemTextColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::ListGetFirstSelected

int ListGetFirstSelected ( )

Retrieves the row index of the first selected item, otherwise -1.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | ListGetItemState | ListGetNextSelected | ListSelectItem

CTreeListCtrl::ListGetFocusItem

int ListGetFocusItem ( )

Retrieves the row index of the first selected item, otherwise -1.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFocusColumn | GetFocusItem | ListSelectItem | ListSetItemState

CTreeListCtrl::ListGetItemBkColor

COLORREF ListGetItemBkColor (int iRow , int nCol = 0) const

Retrieves the background color of an item.

iRow	Index of the row in the tree control with the item to retrieve the color.
nCol	Is the column of the item.

Returns the RGB value of the background color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | ListGetItemTextColor | ListSetColor | ListSetItemBkColor | ListSetItemTextColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::ListGetItemCheckBox

BOOL ListGetItemCheckBox (int iRow , int nCol = 0 , UINT uMask = 0x0F)

Gets the TVIS_STATEIMAGEMASK bits from an item in a row, for the button state.

iRow	Is the row of the item.
nCol	Is the colomn of the button.
uMask	Selects wich bits in the TVIS_STATEIMAGEMASK should be changed.

Returns the button state:

If TVS_EX_BITCHECKBOX isn't enabeld:
0 = not visible
1 = not selected
2 = selected
If TVS_EX_BITCHECKBOX is enabeld:
0 = not selected
1 = selected

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemCheckBox | ListGetItemState | ListSetItemState | SetItemCheckBox | ListSetItemCheckBox

CTreeListCtrl::ListGetItemImage

int ListGetItemImage (int iRow , int &nImage , int &nSelectedImage)

Retrieves the image of an entry.

iRow	Index of the row in the tree control with the item to retrieve the color.
nImage	An integer that receives the index of the item's image within the tree view control's image list.
nSelectedImage	An integer that receives the index of the item's selected image within the tree view control's image list.

Returns nonzero if successful; otherwise 0..

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItem | SetItemImageEx

CTreeListCtrl::ListGetItemImageEx

BOOL ListGetItemImageEx (int iRow , int *pImage , int nCol = 0)

Retrieves the image of an entry.

iRow	Row index of the item whose image is to be retrieved.
pImage	Pointer whose retrieves the image number, or -1 if no image is asserted to the item.
nCol	Is the column of the item.

Returns TRUE if the image number was detected or FALSE if an error occurs.

int GetItemImageEx (HTREEITEM hItem , int nCol = 0)

Retrieves the image of an entry.

iRow	Row index of the item whose image is to be retrieved.
nCol	Is the column of the item.

Returns the image number or -1 no image is asserted to the item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItem | SetItemImageEx

CTreeListCtrl::ListGetItemRect

BOOL ListGetItemRect (int iRow , RECT *pRect , int iCode)
BOOL ListGetItemRect (int iRow , int nCol , RECT *pRect , int iCode)

Call this function to retrieve the bounding rectangle for *an* item and determine whether it is visible or not.

iRow	Row index of the item whose bounding rectangle is to be retrieved.
nCol	Selects the column from which the rectangle should be retrieved. If TVIR_GETCOLUMN in iCode insn't set, the full row is used to get the rectangle.

pRect	Pointer to a RECT structure that receives the bounding rectangle. The coordinates are relative to the upper-left corner of the tree view control.
iCode	This flags define which part of the item should be retrieved. TVIR_TEXT : retrieve only the text rectangle TVIR_GETCOLUMN : retrieve only the column rectangle

Returns nonzero if the item is visible, with the bounding rectangle contained in *lpRect*. Otherwise, 0 with *lpRect* uninitialized.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EnsureVisible | GetItemRect | ListEnsureVisible

CTreeListCtrl::ListGetItemState

UINT ListGetItemState (int iRow , UINT nStateMask , int nCol = 0)

Retrieve the state of an item.

iRow	Row index of the item whose state bits are to be retrieved.
nStateMask	Mask indicating which states are to be retrieved:

TVIS_BOLD	The item is bold.
TVIS_CUT	The item is selected as part of a cut-and-paste operation.
TVIS_DROPHILITED	The item is selected as a drag-and-drop target.
TVIS_EXPANDED	The item's list of child items is currently expanded; that is, the child items are visible. This value applies only to parent items. This bits is only for the first column.
TVIS_EXPANDEDONCE	The item's list of child items has been expanded at least once. The TVN_ITEMEXPANDING and TVN_ITEMEXPANDED notification messages are not generated for parent items that have this state set in response to a TVM_EXPAND message. Using TVE_COLLAPSE and TVE_COLLAPSERESET with TVM_EXPAND will cause this state to be reset. This value applies only to parent items. This bits is only for the first column.
TVIS_EXPANDPARTIAL	A partially expanded tree list item. In this state, some, but not all, of the child items are visible and the parent item's plus symbol is displayed. This bits is only for the first column.
TVIS_SELECTED	The item is selected. Its appearance depends on whether it has the focus. The item will be drawn using the system colors for selection. This bits is only for the first column.
TVIS_OVERLAYMASK	Mask for the bits used to specify the item's overlay image index.
TVIS_STATEIMAGEMASK	Mask for the bits used to specify the item's state image index. This bits are used to get the check box state too.
TVIS_USERMASK	Same as TVIS_STATEIMAGEMASK.
TVIS_UNTERLINE	The item text is underlined.

nCol	Selects the column of the item entry.

Returns the state bits of the selected item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | ListGetItemText | ListSetItemState | ListSetItemText | SetItemState

CTreeListCtrl::ListGetItemText

BOOL ListGetItemText (int iRow , LPTSTR pBuffer , int iMax , int nCol)
LPCTSTR ListGetItemText (int iRow , int nCol)

Gets the text of an item and stores it in an buffer, or retrieve a pointer to the text.

iRow	Row index of the item whose text is to be retrieved.
pBuffer	Is the text buffer where the text will be saved.
iMax	Is the size of the text buffer in chars.

nCol	Is the column of the item.

Returns a pointer to the text.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | ListGetItemState | ListSetItemState | ListSetItemText | SetItemState

CTreeListCtrl::ListGetItemTextColor

COLORREF ListGetItemTextColor (int iRow , int nCol) const

Retrieves the text color of an item.

iRow	Row index of the item whose text color is to be retrieved.
nCol	Selects the column of the item raw.

Returns the RGB value of the text color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | ListGetItemBkColor | ListSetColor | ListSetItemBkColor | ListSetItemTextColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::ListGetNextSelected

int ListGetNextSelected (int iRow)

Retrieves the row index of the next selected item.

iRow	Is the handle of the item, where the search should be started. Use **TVI_ROOT** to get the first selected item.

Returns the row index of the next selected iten, or -1 if no other selected item is present.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | ListGetFirstSelected | ListGetItemState | ListSelectItem

CTreeListCtrl::ListGetTopIndex

int ListGetTopIndex ( )

Returns the index of the topmost visible item.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetRowCount | GetRowOfItem | GetCountPerPage | ListSetTopIndex

CTreeListCtrl::ListGetUserData

LPVOID ListGetUserData (int iRow) const

Retrieves the pointer to the user data field of a item. The user data field is an area which is added automatically to each item. The field has the same size for each item. The size of the field could be defined with the SetUserDataSize function.

iRow	Row index of the item whose user data pointer is to be retrieved.

Returns a pointer to the user data, otherwise NULL.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetUserData | GetUserDataSize | SetUserDataSize

CTreeListCtrl::ListInsertItem

int ListInsertItem (int iRow , LPCTSTR pText , int nImage , int nState , int iMask)

Inserts an item in a Tree-List-Control which is used as List-Control.

iRow	Is the row index where the item should be inserted.
pText	Is the text for the item.
nImage	Is the number for the icon. Use TV_NOIMAGE for no icon.
nState	Is the state of the image. TVIS_BOLD = text is bolded TVIS_UNTERLINE = text is underlined TVIS_SELECTED = item is selected TVIS_OVERLAYMASK = overlay bits for image TVIS_STATEIMAGEMASK = image for state icons
iMask	Is the mask of bits which are used in the nState parameter TVIS_BOLD = text is bolded TVIS_UNTERLINE = text is underlined TVIS_SELECTED = item is selected TVIS_OVERLAYMASK = overlay bits for image TVIS_STATEIMAGEMASK = image for state icons

Returns the insert position of the item or -1 if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | ListDeleteItem | ListSetItem

CTreeListCtrl::ListSelectDropTarget

BOOL ListSelectDropTarget (int iRow , int nCol)

Call this function to redraw the item in the style used to indicate the target of a drag-and-drop operation.

iRow	Is the row index of a tree item.
nCol	Is the column of the item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetDropHilightColumn | ListCreateDragImage | SelectDropTarget

CTreeListCtrl::ListSelectItem

BOOL ListSelectItem (int iRow , int nCol)

Call this function to select the given tree view item. If *iRow* is below zero, then this function selects no item.

iRow	Is the row index of a tree item.

nCol	Is the column of a tree item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | ListGetFirstSelected | ListGetItemState | ListGetNextSelected

CTreeListCtrl::ListSetColor

BOOL ListSetColor (int iRow , int iCol , COLORREF uBkColor , COLORREF uTextColor)
BOOL ListSetColor (int iRow , int iCol , COLORREF uBkColor = TV_NOCOLOR , COLORREF uTextColor = TV_NOCOLOR)

Changes the colors of an sItem in a Tree-List-Control which is used as List-Control

iRow	Is the row index of the item.
iCol	Is the column of the item.
uBkColor	Is the new background color. Use TV_NOCOLOR to set the default background color of the control.
uTextColor	Is the new text color. Use TV_NOCOLOR to set the default text color of the control.

Returns TRUE if success full, otherwise FALSE.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | ListGetColor | ListGetItemBkColor | ListGetItemTextColor | ListSetItemBkColor | ListSetItemTextColor | SetColor

CTreeListCtrl::ListSetFocusItem

BOOL ListSetFocusItem (int iRow , int nCol)

Selects the item with the focus.

iRow	Is the row index of the item.
nCol	Is the column wich receives the focus. -1 means no change.

Returns TRUE if the focus was selected, FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetFirstSelected | GetFocusColumn | GetFocusItem | SelectItem

CTreeListCtrl::ListSetItem

BOOL ListSetItem (int iRow , LPCTSTR pText , int nImage = -1 , int nState = -1 , int iMask = TVIS_BOLD|TVIS_UNTERLINE|TVIS_SELE
BOOL ListSetItem (int iRow , int iCol , LPCTSTR pText , int nImage = -1 , int nState = -1 , int iMask = TVIS_BOLD|TVIS_UNTERLINE|TVIS_SELE(

Changes an item in a Tree-List-Control which is used as List-Control.

iRow	Is the row index of the item.
iCol	Is the column of the item.

pText Is the text for the item.

nImage Is the number for the image.
Use TV_NOIMAGE to remove the image.

nState Is the state of the image:

TVIS_BOLD	text is bolded
TVIS_UNTERLINE	text is underlined
TVIS_SELECTED	item is selected
TVIS_OVERLAYMASK	overlay bits for image
TVIS_STATEIMAGEMASK	image for state icons (only for column 0)

iMask Is the mask of bits which are used in the *nState* parameter.

Returns TRUE if ok or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | ListDeleteItem | ListInsertItem | ListSetItemBkColor | ListSetItemState | ListSetItemText | ListSetItemTextColor

CTreeListCtrl::ListSetItemBkColor

COLORREF ListSetItemBkColor (int iRow , int nCol , COLORREF uColor)

Changes the background color of an item.

iRow	Is the row index of the item.
nCol	Is the column of the item.
uColor	Is the RGB value of the background color. Use TV_NOCOLOR to set the default background color of the control.

Returns the new RGB color value of the item, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | ListGetItemBkColor | ListGetItemTextColor | ListSetColor | ListSetItemTextColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::ListSetItemCheckBox

BOOL ListSetItemCheckBox (int iRow , int iState , int nCol , UINT uMsk)

Sets the TVIS_STATEIMAGEMASK bits in an item, for the button state.

iRow	Is the handle of the Item.
iState	Is the new state of the button: 0 = not visible 1 = not selected 2 = selected If TVS_EX_BITCHECKBOX is enabeld: 0 = not selected 1 = selected
nCol	Is the colomn of the button.
uMsk	Selects wich bits in the TVIS_STATEIMAGEMASK should be changed.

Returns nonzero if successful, otherwise zero.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemCheckBox | ListGetItemState | ListSetItemState | SetItemCheckBox | ListGetItemCheckBox

CTreeListCtrl::ListSetItemImage

BOOL ListSetItemImage (int iRow , int nImage , int nSelectedImage)

Call this function to set the index of the item's image and its selected image within the tree view control's image list.

iRow	Row index of the item whose image is to be set.
nImage	Index of the item's image in the tree view control's image list.
nSelectedImage	Index of the item's selected image in the tree view control's image list.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItem | SetItemImageEx

CTreeListCtrl::ListSetItemImageEx

BOOL ListSetItemImageEx (int iRow , int nImage , int nCol = 0)

Call this function to set the index of the item's image within the tree list view control's image list.

iRow	Row index of the item whose image is to be set.
nImage	Index of the item's image in the tree view control's image list.
nCol	Index of the item's selected image in the tree view control's image list.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemImageEx | SetItem | SetItemImageEx

CTreeListCtrl::ListSetItemState

BOOL ListSetItemState (int iRow , UINT nState , UINT nStateMask)
BOOL ListSetItemState (int iRow , int nCol , UINT nState , UINT nStateMask)

Sets the state of the item specified by a row.

iRow	Row index of the item whose state is to be set.
nCol	Is the column of the item.
nState	Specifies new states for the item. For information on states, see CTreeListCtrl::GetItemState.
nStateMask	Specifies which states are to be changed.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | ListGetItemState | ListGetItemText | ListSetItemText | SetItemState

CTreeListCtrl::ListSetItemText

BOOL ListSetItemText (int iRow , LPCTSTR pText , int nCol)

Sets the text of the item specified by a row.

iRow	Row index of the item whose text is to be set.
pText	Address of a string containing the new text for the item.
nCol	Is the column of the item.

Returns nonzero if successful, otherwise 0.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemState | ListGetItemState | ListGetItemText | ListSetItemState | SetItemState

CTreeListCtrl::ListSetItemTextColor

COLORREF ListSetItemTextColor (int iRow , int nCol , COLORREF uColor)

Retrieves the background color of an item.

iRow	Index of the row in the tree control with the item to set the color.
uColor	Is the RGB value of the text color. Use **TV_NOCOLOR** to select the default text color.
nCol	Is the column of the item.

Returns the RGB value of the text color, or **TV_NOCOLOR** if the color is the default color of the control.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemBkColor | GetItemTextColor | ListGetItemBkColor | ListGetItemTextColor | ListSetColor | ListSetItemBkColor | SetItemBkColor | SetItemTextColor

CTreeListCtrl::ListSetTopIndex

BOOL ListSetTopIndex (int iRow)

Scrolls to the item specified by an index at the top of the view.

iRow	Specifies the zero-based index of the list-box item.

Returns TRUE if success full, or FALSE if an error occurs.

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetCountPerPage | ListGetTopIndex

CTreeListCtrl TVN_CBSTATECHANGED

This notify message will be send to the parent window if the state of an auto edit column item was changed. For the auto edit column the modes **TVAE_CHECK** and **TVAE_CHECKED** must be used.

```
typedef struct
    {
    NMHDR     hdr;
```

```
UINT     action;
TVITEM   itemOld;
TVITEM   itemNew;
POINT    ptDrag;
}NM_TREEVIEW;
```

hdr — Is the notify header.

hwndFrom	is the caller window handle
idFrom	is the ID from the caller window
code	is **TVN_CBSTATECHANGED**

action — Is reason for the edit action.

VK_EDITCLK	a single click on the selected item
VK_DBLCLK	a double click on an item
VK_RETURN	enter was pressed for an item
<char>	char input on the selected item

Do allow **VK_EDITCLK** the extended style **TVS_EX_EDITCLICK** must be set.

itemOld — Isn't used.

itemNew — Contains the new state of the item for in the *state* member. The bits 0xF000 contains the new state, if the extended state **TVS_EX_BITCHECKBOX** is set. The bit 0x1000 contains the new state.

The *cChildren* member contains the column number.

For more details look at CTreeCtrl::SetItem in the MFC documentaion.

ptDrag — Contains the mouse coortinates if **VK_EDITCLK** or **VK_DBLCLK** action.

If the return value is ignored.

```
ON_NOTIFY(TVN_CBSTATECHANGED,IDC_TREELIST,OnChangeNotify)

...

void CMyDialog::OnChangeNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
NM_TREEVIEW *pNmTreeView = (NM_TREEVIEW*)pNmHdr;


    if(pNmTreeView->itemNew.state&0x1000)
          ChangedTo(0);
    else    ChangedTo(1);

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumnAutoEdit | Extended Styles

CTreeListCtrl TVN_COLUMNCHANGED

This notify message will be send if the size of a column item will be changed. This message will be sended only if the extended style **TVS_EX_HEADERCHGNOTIFY** is enabled.

```
typedef struct
    {
    NMHDR    hdr;
    UINT     uColumn;
    UINT     uIndex;
    UINT     uPosX;
    INT      iSize;
    }TV_COLSIZE;
```

hdr — Is the notify header.

hwndFrom	is the caller window handle
idFrom	is the ID from the caller window
code	is **TVN_COLUMNCHANGED**

uColumn — Is the item index of the changed column.

uIndex — Is the visible index of the column.

uPosX — Is the X position of the column in the header control.

iSize	Is the new size of the column.

```
ON_NOTIFY(TVN_COLUMNCHANGED,IDC_TREELIST,OnChangeColumnNotify)

...

void CMyDialog::OnChangeColumnNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
TV_COLSIZE    *pNotify = (TV_COLSIZE*)pNmHdr;
int            iCol;


    iCol = pNotify->uColumn;

    ...

    *pResult = 0;

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members

CTreeListCtrl TVN_ENDLABELEDIT

This notify message is called if a label edit was finished.

```
typedef struct
    {
    NMHDR     hdr;
    TVITEM    item;
    }NMTVDISPINFO;
```

hdr	Is the notify header.	
	hwndFrom	is the caller window handle
	idFrom	is the ID from the caller window
	code	is **TVN_ENDLABELEDIT**
item	Contains the new text of the item in the *pszText* member. In *cchTextMax* the size of the text buffer is stored. If the ESC key was pressed the buffer size is zero. In the other case the user can change this text. If the text was changed the **TVIF_TEXTCHANGED** flag is set in the *mask* member. If the *return* key was pressed the **TVIF_RETURNEXIT** flag is set in the *mask* member. If the *ESC* key was pressed the **TVIF_CANCELED** flag is set in the *mask* member. The *cChildren* member contains the column number. For more details look at CTreeCtrl::SetItem in the MFC documentaion.	

If the return value is zero the new text will be stored in item.

```
ON_NOTIFY(TVN_ENDLABELEDIT,IDC_TREELIST,OnEndLabelEdit)

...

void CMyDialog::OnEndLabelEdit(NMHDR *pNmHdr,LRESULT *pResult)
{
NMTVDISPINFO *pNmTreeView = (NMTVDISPINFO*)pNmHdr;


    if(pNmTreeView->item.cchTextMax>1)         // Change text
        {
        pNmTreeView->item.pszText[0] = 'X';
        }

    if(pNmTreeView->item.mask&TVIF_RETURNEXIT)
        {
        // the user has pressed RETURN to input the text
        }

    if(pNmTreeView->item.mask&TVIF_TEXTCHANGED)
```

```
        {
        TextWasChanged();
        }

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumnAutoEdit | EditLabel | EditLabelCb

CTreeListCtrl TVN_ITEMTOOLTIP

This notify message will be send to the parent window if the mouse cursor moves over an item. In the response of the massege the parent can change the text, position, and the delay to showing the tooltip window.

If the return value of the message is zero, the common tooltip text will be shown, if the text is greater than the coloumn.

For this message the extended style **TVS_EX_TOOLTIPNOTIFY** must be set.

```
        typedef struct
            {
            NMHDR     hdr;
            UINT      action;
            TVITEM    itemOld;
            TVITEM    itemNew;
            POINT     ptDrag;
            }NM_TREEVIEW;
```

hdr	Is the notify header. hwndFrom is the caller window handle idFrom is the ID from the caller window code is **TVN_ITEMTOOLTIP**
action	Allways zero.
itemOld	Isn't used.
itemNew	Contains the text for the tooltip window. The text must stored in the *pszText* member. Use an own buffer, don't overwrite the original text. If you want a delayed viewing of the tooltip, set the **TVIF_TOOLTIPTIME** flag in the *mask* member, and set the delay time in the *lParam* member. (in milliseconds) The *cChildren* member contains the column number. For more details look at CTreeCtrl::SetItem in the MFC documentaion.
ptDrag	Contains the top left position of thr item text, and is used as the position for the tooltip. The usercan change the position to move the tool tip.

If the return value is zero the common tooltip will be shown.
In the other case the user definded tooltip will be shown.

```
ON_NOTIFY(TVN_ITEMTOOLTIP,IDC_TREELIST,OnTooltipNotify)

...

void CMyDialog::OnTooltipNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
NM_TREEVIEW *pNmTreeView = (NM_TREEVIEW*)pNmHdr;
                                        // Is the text for the tootip (""=no Tooltip)
   pNmTreeView->itemNew.pszText= "User definded\nTooltip\nMessage";
   pNmTreeView->ptDrag.x      += 20;    // Move Tooltip to an other position
   pNmTreeView->ptDrag.y      += 20;
   pNmTreeView->itemNew.mask  |= TVIF_TOOLTIPTIME; // Set a delay to show the tooltip
   pNmTreeView->itemNew.lParam = 1000;

   *pResult = 1;                        // 0=Common-Tooltip 1=User-Tooltip

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | GetItemText | GetItemRect | Extended Styles

CTreeListCtrl TVN_STARTEDIT

This notify message will be send if a lebel edit acion could be taken.
The receiver decides with the return value if an edit action should be started.

```
typedef struct
    {
    NMHDR     hdr;
    TVITEM    item;
    UINT      uAction;
    UINT      uHeight;
    UINT      uMaxEntries;
    LPCTSTR   pTextEntries;
    LPCTSTR *pTextList;
    POINT     ptAction;
    }TV_STARTEDIT;
```

hdr — Is the notify header.

hwndFrom	is the caller window handle
idFrom	is the ID from the caller window
code	is **TVN_STARTEDIT**

item — Is the item data. The *cChildren* member contains the column number.
For more details look at CTreeCtrl::SetItem in the MFC documentaion.

uAction — Is reason for the edit action.

VK_EDITCLK	a single click on the selected item
VK_DBLCLK	a double click on an item
VK_RETURN	enter was pressed for an item
<char>	char input on the selected item

Do allow **VK_EDITCLK** the extended style **TVS_EX_EDITCLICK** must be set.

uMaxEntries — Maximum count of drop list items in the used combobox.

pTextEntries — Is a string with substrings, which are seperated by special signs, for the drop list items in the used combobox. The sign must be selected in the return value with the **TVIR_EDITCOMBOCHAR(c)** macro.

i.e.:

"Abc|Xyz|Lmn" for **TVIR_EDITCOMBOCHAR('|')**
"Abc\0Xyz\0Lmn" for **TVIR_EDITCOMBOCHAR(0)** or none.

pTextList — Is a field of strings for the drop list items in the used combobox.
The last string must be zero or *uMaxEntries* must be set.

i.e.:

```
LPCTSTR  field[3];

field[0] = "Abc";
field[1] = "Xyz";
field[2] = 0;
```

ptAction — Contains the mouse coortinates if **VK_EDITCLK** or **VK_DBLCLK** action.

If the return value is zero no label edit will be started.

This are the allowed flags in the return value:

TVIR_EDITTEXT	edit the text with an edit window
TVIR_EDITFULL	stretch the edit window over the full item rectangle
TVIR_EDITCOMBOBOX	edit the text with an combobox
TVIR_EDITCOMBOCHAR(n)	selects the seperator sign for *pTextEntries*
TVIR_EDITCOMBODOWN	show the drop down list of the combobox
TVIR_EDITCOMBODEL	deletes the *pTextEntries* buffer with *delete*
TVIR_EDITCOMBOLIST	the combobox has only a down list and no edit field

```
ON_NOTIFY(TVN_STARTEDIT,IDC_TREELIST,OnStartEditNotify)

...

void CMyDialog::OnStartEditNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
TV_STARTEDIT *pNotify = (TV_STARTEDIT*)pNmHdr;
int           iCol;


    iCol = pNotify->item.cChildren;

    if(iCol>=4)                  // Don't edit this columns
```

```
        {
        *pResult = 0;
        return;
        }

    if(iCol==0)                  // Use an edit window
        {
        *pResult = TVIR_EDITTEXT;
        return;
        }

    if(iCol==1)                  // Use a combobox (list)
        {
        LPCTSTR  aField[3];

        aField[0] = "Abc";
        aField[1] = "Xyz";
        aField[2] = 0;

        pNotify->pTextList = aField;

        *pResult = TVIR_EDITCOMBOBOX|TVIR_EDITCOMBOLIST;
        return;
        }

    if(iCol==2)                  // Use a combobox (edit)
        {
        LPCTSTR  aField[2];

        aField[0] = "Abc";
        aField[1] = "Xyz";

        pNotify->pTextList   = aField;
        pNotify->uMaxEntries = 2;

        *pResult = TVIR_EDITCOMBOBOX|TVIR_EDITFULL;
        return;
        }


    pNotify->pTextEntries    = "Abc|Xyz|Lmn";
    *pResult = TVIR_EDITCOMBOBOX|TVIR_EDITCOMBOCHAR('|');

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | EditLabel | EditLabelCb | SetColumnAutoEdit

CTreeListCtrl TVN_STEPSTATECHANGED

This notify message will be send to the parent window if the state of an auto edit column item was changed. For the auto edit column the modes **TVAE_STEP** and **TVAE_STEPED** must be used.

```
        typedef struct
            {
            NMHDR    hdr;
            UINT     action;
            TVITEM   itemOld;
            TVITEM   itemNew;
            POINT    ptDrag;
            }NM_TREEVIEW;
```

hdr — Is the notify header.

hwndFrom	is the caller window handle
idFrom	is the ID from the caller window
code	is **TVN_STEPSTATECHANGED**

action — Is reason for the edit action.

VK_EDITCLK	a single click on the selected item
VK_DBLCLK	a double click on an item
VK_RETURN	enter was pressed for an item
<char>	char input on the selected item

Do allow **VK_EDITCLK** the extended style **TVS_EX_EDITCLICK** must be set.

itemOld — Isn't used.

itemNew — Contains the new state of the item for in the *state* member. The bits 0xF000 contains the new state, if the extended state **TVS_EX_BITCHECKBOX** is set. The bit 0x1000 contains the new state.

The *cChildren* member contains the column number.

For more details look at CTreeCtrl::SetItem in the MFC documentaion.

ptDrag — Contains the mouse coortinates if **VK_EDITCLK** or **VK_DBLCLK** action.

If the return value is ignored.

```
ON_NOTIFY(TVN_STEPSTATECHANGED,IDC_TREELIST,OnChangeNotify)

...

void CMyDialog::OnChangeNotify(NMHDR *pNmHdr,LRESULT *pResult)
{
NM_TREEVIEW *pNmTreeView = (NM_TREEVIEW*)pNmHdr;


    if(pNmTreeView->itemNew.state&0x1000)
            ChangedTo(0);
    else    ChangedTo(1);

}
```

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | SetColumnAutoEdit | Extended Styles

CTreeListCtrl::CollapseAll

BOOL CollapseAll (HTREEITEM hItem , int iCode)

Collapses an entry and all his child items.

hItem	Handle of the entry
iFlags	Several options • **TVE_ONLYCHILDS** collapse only the childs of the entry. • **TVE_EXPANDNEXT** start at the parent of the entry

Returns TRUE if succefull

see also:

CTreeListCtrl Overview | CTreeListCtrl Class Members | Expand | ExpandAll | GetChildItem | GetNextItem