Classic Component Set Help Contents

TcsNotebook Component

Properties Methods Events

Unit

CSNoteBk

Description

The TcsNotebook component is similar to the TTabbedNotebook component but differs in that you can:

- Attach the tabs to the top, bottom, left or right of the notebook using the TabOrientation property.
- Specify a different bitmap to be displayed on each tab by editing the Bitmap property.
- Specify how many glyphs are in the chosen Bitmap by editing the NumGlyphs property.
- Disable specific notebook pages by editing the Pages property.
- Specify the color of the selected tab's caption text using the SelectedColor and the color of the unselected tab's caption text using the UnselectedColor.
- Specify the color of the notebook and the selected tab using the Color and ParentColor properties.
- Specify the color of the unselected tabs using the UnselectedTabColor and UseUnselectedTabColor properties.
- Specify the alignment of the tab’s caption text using the TextAlignment property.
- Specify the alignment of the tab’s bitmap using the BitmapAlignment property.
- Save resources by releasing the handles for the controls on the current page when another page is selected by setting the SaveResources property.
- Control the appearance of the tabs/notebook with the CornerSize, MaxTabWidth, RowIndent, SidewaysText, TabHeight properties.
- Change the behaviour of the notebook with the AnchoredTabs property.
- Easily access the Notebook's Pages property at design time by using the (right click) context menu.

Properties

ActivePage
AnchoredTabs
Bitmap
BitmapAlignment
CornerSize
MaxTabWidth
NumGlyphs
PageIndex
Pages
ParentTabFont
RowExtent
RowIndent
SaveResources
SelectedColor
SidewaysText
TabFont
TabHeight
TabOrientation
TabsPerRow
TextAlignment
UnselectedColor
UnselectedTabColor
UseUnselectedTabColor

Run-time only:

TabBitmap
TabCaption
TabNumGlyphs
TabPageEnabled
TabPageIndex
TabPageVisible

Events

OnPageChanged
OnPageChanging
OnTabClick

Methods

TabAtPos

CSNoteBk Unit

The CSNoteBk unit contains the declaration for the TcsNotebook component and its associated objects.

The following items are declared in the CSNoteBk unit:

Components

TcsNotebook

TcsPage

Types

TTabNumGlyphs

TTabOrientation

TTextAlignment

TBitmapAlignment

TTabRects = (trAll, trSelected, trUnselected);

TcsTabPoints = Array[0..5] of TPoint;

TPageChangingEvent

ActivePage Property

Applies to

TcsNotebook, TNotebook, TTabbedNotebook component

Declaration

property ActivePage: **String**;

Description

The ActivePage property determines which page is displayed in the notebook or tabbed notebook control. The value of ActivePage must be one of the strings contained in the Pages property.

AnchoredTabs Property

Applies to

TcsNotebook component

Declaration

property AnchoredTabs: Boolean;

The AnchoredTabs property determines whether tabs will remain in place when selected or will be brought to the front row. The default value is False.

Bitmap Property

Applies to

TcsNotebook, TcsPage component

Declaration

property Bitmap: TBitmap;

Description

The Bitmap property for TcsNotebook indicates the bitmap for the currently selected page. The Bitmap property for TcsPage indicates the bitmap for that page's tab. The specified bitmap can contain multiple glyphs, according to NumGlyphs, which are used for the different states of the page's tab. Note that each page has its own Bitmap and NumGlyphs properties. To easily assign bitmaps to each tab, select the TcsNotebook component on the form, choose the Bitmap property in the Object Inspector and then use the context menu (right-click the component) whenever you need to assign the Bitmap for a different page.

BitmapAlignment Property

Applies to

TcsNotebook component

Declaration

property BitmapAlignment: TBitmapAlignment;

Description

The BitmapAlignment property determines the placement of the appropriate bitmap (if specified) on each tab. Note that the different bitmap alignment options are always regarded as though you are viewing the tab in Top orientation, doing the stated bitmap alignment, and then rotating the tab (but not rotating the bitmap).

TBitmapAlignment Type

Unit

CSNoteBk

Declaration

TBitmapAlignment = (baLeftTop, baCentreTop, baCenterTop, baRightTop, baLeftMiddle, baCentreMiddle, baCenterMiddle, baRightMiddle, baLeftBottom, baCentreBottom, baCenterBottom, baRightBottom, baFit, baInvisible);

Description

TBitmapAlignment defines the possible values the BitmapAlignment property of a TcsNotebook object can assume.

CornerSize Property

Applies to

TcsNotebook component

Declaration

property CornerSize: Integer;

Description

The CornerSize property can be used to alter the size of the corner of each Tab. The value specified is the amount, in pixels, which is to be cut' off the corner of the tab. The CornerSize will always be greater than 0. Specify a CornerSize value of 1 if you want Win95 style (square) tabs.

Note that in certain situations the control will automatically adjust the CornerSize to prevent inappropriate values relative to the values of the following properties: MaxTabWidth, TabHeight.

MaxTabWidth Property

Applies to

TcsNotebook component

Declaration

property MaxTabWidth: Integer;

Description

The MaxTabWidth property controls the maximum width, in pixels, of each tab. When set to 0 (the default) the tabs in a row will be sized to spread evenly across the full width of each row. If you don't want the tabs to ever be wider than a certain amount, for example if each tab has a short caption, you should change MaxTabWidth to the desired value. Note that you are not specifying the tab's width but the maximum width it can be. If the tab is narrower than MaxTabWidth (by virtue of the width of the notebook and how many tabs there are per row) the tab's width will not be increased to MaxTabWidth.

Increasing the TabsPerRow property to a value greater than the actual no. of tabs will have a similar effect to increasing MaxTabWidth but with all tabs always in a single row.

Note that in certain situations the control will automatically adjust the MaxTabWidth to prevent inappropriate values relative to the value of the CornerSize property.

NumGlyphs Property

Applies to

TcsNotebook, TcsPage components

Declaration

property NumGlyphs: TTabNumGlyphs;

Description

The NumGlyphs property for TcsNotebook indicates the number of glyphs in the Bitmap for the current tab. The NumGlyphs property for TcsPage indicates the number of glyphs in the Bitmap for that page's tab. Each page has its own NumGlyphs property. The default value is 1. The first glyph is used for the Selected tab state, the second glyph (if NumGlyphs is > 1) is used for the Disabled tab state and the third glyph (if NumGlyphs > 2) is used for the Unselected tab state.

TTabNumGlyphs Type

Unit

CSNoteBk

Declaration

TTabNumGlyphs = 1..3; *{ Selected, Disabled, Unselected }*

Description

The TTabNumGlyphs type defines the range of values the NumGlyphs property of TcsNotebook or TcsPage can assume.

PageIndex Property

Applies to

TcsNotebook, TNotebook, TTabbedNotebook component

Declaration

property PageIndex: Integer;

Description

The PageIndex property determines the current page in the notebook. Changing the PageIndex value changes the page currently displayed by the control. The PageIndex value is zero based and will range from 0 up to the number of tabs minus 1, i.e. 0..4 if there are 5 tabs. Each new page added will be given the next available page number. Note that if you delete pages the PageIndex values will be re-assigned to the remained tabs, starting from 0 and increasing.

You can also change the current page by using the ActivePage property.

Pages Property

See also Bitmaps

Applies to

TcsNotebook, TNotebook, TTabbedNotebook component

Declaration

property Pages: TStrings;

Description

The Pages property contains the captions that identify each page in the notebook. The control ensures that the captions for each page are unique. Each caption in Pages will have an associated page which contains the controls for that page. Each page can be enabled or disabled, with the appearance of the page's tab being changed accordingly. Pages can also be hidden so that no tab is shown. The bitmap (if specified) for each page is also contained in each page.

To access the actual pages themselves you can use the Objects property of Pages. Each page, i.e. each item in Objects[], is of type TcsPage. To access the tab caption for a page you can also use the TabCaption property. To access the bitmap for a page you can also use the TabBitmap property. To access the NumGlyphs property for a page you can also use the TabNumGlyphs property. To access the PageEnabled property for a page you can also use the TabPageEnabled property. To access the PageVisible property for a page you can also use the TabPageVisible property.

See Also

Add Tab Dialog Box

Edit Tab Dialog Box

TabBitmap

TabCaption

TabNumGlyphs

TabPageEnabled

TabPageIndex

TabPageVisible

TcsNotebook

Bitmaps for TcsNotebook Pages

Normally, the bitmaps for each tab of the TcsNotebook would be specified at design-time using the Bitmap property. However, if you want to assign or change the bitmaps for the tabs at run-time you can do this by using the TabBitmap property. The following examples show how to do this when the form is first created . The examples assume that you have added a TcsNotebook component to Form1 and that it is called csNotebook1. The first example shows how a new bitmap can be assigned to the first tab (page 0) by loading a bitmap file.

Example 1

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  csNotebook1.TabBitmap[0].LoadFromFile('c:\bitmaps\map.bmp');
end;
```

The next example shows how a bitmap which has been included in a resource file can be assigned to the first tab. The bitmap's resouce identifier is 'mybitmap'.

Example 2

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  csNotebook1.TabBitmap[0].Handle :=
    LoadBitmap(HInstance, 'mybitmap');
end;
```

ParentTabFont Property

Applies to

TcsNotebook component

Declaration

property ParentTabFont: Boolean;

Description

The ParentTabFont property determines where the notebook looks for the font information for the tabs. If ParentTabFont is True, the notebook uses the font in its parent component's Font property for the tabs. If ParentTabFont is False, the notebook uses the TabFont property for the tabs.

Specifying a different TabFont value will automatically set ParentTabFont to False. Setting ParentTabFont to True automatically sets the notebook's TabFont to the same font as its parent component's Font property.

RowExtent Property

Applies to

TcsNotebook component

Declaration

property RowExtent: Integer;

Description

The RowExtent property is a calculated read-only property which specifies the number of rows of tabs currently displayed. It is recalculated if the number of tabs is changed or if the TabsPerRow property is changed.

RowIndent Property

Applies to

TcsNotebook component

Declaration

property RowIndent: Integer;

Description

The RowIndent property is the indent, in pixels, added to each successive row of tabs after the first.

SaveResources Property

Applies to

TcsNotebook component

Declaration

property SaveResources: Boolean;

Description

The SaveResources property indicates whether resources will be saved by releasing the Windows handles for the components on a page when another page is selected. Normally, the handles for the components on a Notebook page are created when the page is first shown and then only released when the form is destroyed. If SaveResources is True then the handles for the components on a page will be released when another page is selected, thus reducing the total resources required.

Note that if SaveResources is True there may be a slight increase in how long it takes to display a new page (even if it has previously been selected) because the handles must be re-created each time the page is re-selected.

SelectedColor Property

Applies to

TcsNotebook component

Declaration

property SelectedColor: TColor;

Description

The SelectedColor property indicates the color to be used for the text of the selected tab, the default color is clBtnText.

SidewaysText Property

Applies to

TcsNotebook component

Declaration

property SidewaysText: Boolean;

Description

The SidewaysText property indicates whether tab text is to be displayed sideways when using Left or Right tab orientation. You can only set SidewaysText to True when TabOrientation is toLeft or toRight. Changing TabOrientation from toTop or toBottom to toLeft or toRight will automatically set SidewaysText to True. Changing TabOrientation from toLeft or toRight to toTop or toBottom will automatically set SidewaysText to False. When SidewaysText is True the list of fonts available when editing the TabFont property only includes TrueType fonts so that text rotation will be possible.

TabFont Property

Applies to

TcsNotebook, TTabbedNotebook component

Declaration

property TabFont: TFont;

Description

The TabFont property determines the font used on the tabs of the notebook control. The text of the selected tab is boldfaced if the selected font for the TabFont property is not also boldfaced.

When using Left or Right oriented tabs (see TabOrientation) and SidewaysText you should use a TrueType font to get rotated text.

TabHeight Property

Applies to

TcsNotebook component

Declaration

property TabHeight: Integer;

Description

The TabHeight property determines the height, in pixels, of each tab.

TabOrientation Property

Applies to

TcsNotebook component

Declaration

property TabOrientation: TTabOrientation;

Description

The TabOrientation property determines the orientation of the tabs around the notebook pages. If the SidewaysText property is True the tab text will be rotated when using Left and Right tab orientations. Note that bitmaps are never rotated, regardless of the values of the TabOrientation or BitmapAlignment properties. Changing TabOrientation from toTop or toBottom to toLeft or toRight or from toLeft or toRight to toTop or toBottom will automatically set the SidewaysText property to the appropriate default value.

TTabOrientation Type

Unit

CSNoteBk

Declaration

TTabOrientation = (toTop, toLeft, toBottom, toRight);

Description

TTabOrientation defines the possible values of the TabOrientation property.

TabsPerRow Property

Applies to

TcsNotebook, TTabbedNotebook component

Declaration

property TabsPerRow: Integer;

Description

The TabsPerRow property determines how many tabs will appear in each row of tabs for the TcsNotebook control. The default value is 3. The appropriate number of rows of tabs will be created to allow for all tabs to be displayed using the TabsPerRow value. If TabsPerRow is greater than the actual number of tabs the size of the tabs will be reduced accordingly; this is similar to changing the MaxTabWidth property.

TextAlignment Property

Applies to

TcsNotebook component

Declaration

property TextAlignment: TTextAlignment;

Description

The TextAlignment property determines where the tab's caption text will be placed on the tab. Note that the different text alignment options are always regarded as though you are viewing the tab in Top orientation, doing the stated text alignment, and then rotating the tab.

TTextAlignment Type

Unit

CSNoteBk

Declaration

TTextAlignment = (taLeftTop, taCentreTop, taCenterTop, taRightTop, taLeftMiddle, taCentreMiddle, taCenterMiddle, taRightMiddle, taLeftBottom, taCentreBottom, taCenterBottom, taRightBottom, taInvisible);

Description

TTextAlignment defines the possible values the TextAlignment property of a TcsNotebook object can assume.

UnselectedColor Property

Applies to

TcsNotebook component

Declaration

property UnselectedColor: TColor;

Description

The UnselectedColor property indicates the color to be used for the text of the unselected tabs, the default color is clBtnText.

UnselectedTabColor Property

Applies to

TcsNotebook component

Declaration

property UnselectedTabColor: TColor;

Description

The UnselectedTabColor property indicates the color used for the background of the unselected tabs. The UnselectedTabColor property is only relevant when the UseUnselectedTabColor property is True. All tabs will have the same background color when UseUnselectedTabColor is False.

UseUnselectedTabColor Property

Applies to

TcsNotebook component

Declaration

property UseUnselectedTabColor: Boolean;

Description

The UseUnselectedTabColor property determines whether the value of the UnselectedTabColor property will be used for the background color of the unselected tabs. When UseUnselectedTabColor is False all the tabs will have the same background color regardless of the UnselectedTabColor value. Also note that the UnselectedTabColor property is always ignored if there is more than one row of tabs and AnchoredTabs is True, even if UseUnselectedTabColor is True.

OnPageChanged Event

Applies to

TcsNotebook, TNotebook component

Declaration

property OnPageChanged: TNotifyEvent;

Description

The OnPageChanged event occurs immediately after a new page has been made the current page.

OnPageChanging Event

Example

Applies to

TcsNotebook component

Declaration

property OnPageChanging: TPageChangingEvent;

Description

The OnPageChanging event is sent before a new page is made the current page. Use the OnPageChanging event handler if you want to prevent a change to a new page, for example when certain validation conditions are not met for the current page. To perform processing after changing to a new page use the OnPageChanged event.

OnPageChanging Event Example

The following example shows how to prevent a change to a new page if certain conditions are not met.

```
procedure TForm1.csNotebook1PageChanging(Sender: Tobject;
                  NewIndex: Integer; var AllowChange: Boolean);
begin
  if not PageIsValid(csNotebook1.PageIndex) then
    AllowChange := False;
end;

function TForm1.PageIsValid(Index: Integer): Boolean;
begin
  case Index of
    0: Result := (Length(Name.Text) > 0);
    1: Result := (Length(Address.Text) > 0);
  end;
end;
```

TPageChangingEvent Type

See also

Unit

CSNoteBk

Declaration

TPageChangingEvent = **procedure**(Sender: TObject; NewIndex: Integer;
var AllowChange: Boolean) **of object**;

Description

The TPageChangingEvent type points to a method that is called before a new page is selected. NewIndex indicates the PageIndex value of the page which is about to be selected. The AllowChange variable indicates if it is permissable to select the new page. If AllowChange is False the new page will not be selected.

See Also

OnPageChanging

OnTabClick Event

Applies to

TcsNotebook component

Declaration

property OnTabClick: TNotifyEvent;

Description

An OnTabClick event occurs when the user selects a tab using one of the following techniques:

(1) pressing the primary mouse button while the mouse pointer is over the tab

(2) pressing the appropriate accelerator key for a tab (as for buttons)

(3) using the arrow keys when the tab has a focus rectangle

An OnTabClick event will not occur if the OnPageChanging event handler prevents the change to the new page.

An OnTabClick event will not occur if a tab is selected programmatically, for example when the ActivePage or PageIndex properties are assigned new values in source code. To respond to a page being made current (regardless of whether the user selected it or it was selected in source code) use the OnPageChanged event instead. The OnTabClick event, when it does occur, will occur after the OnPageChanged event.

Technical Support

Technical support for the Classic Component Set can be obtained from Classic Software via:

CompuServe:	100033,1230
Email:	100033.1230@compuserve.com
Phone/Fax:	+61 9 271 5407 (Local time = Greenwich Mean Time + 8 hours)
Mail:	Unit 2, 19A Wood Street Inglewood WA 6052 Australia

For news about forthcoming components and products watch our web page at:
http://ourworld.compuserve.com/homepages/ClassicSoftware

Country Code	Area Code (WA)	Telephone No.
61	9	271 5407

Notebook Tabs Editor

Use the Notebook Tabs editor to add, edit, delete or move tabs in a TcsNotebook component. The editor shows the Caption, Help Context, Enabled and Visible information for each tab. The bitmap for the highlighted row of the list of tabs is also shown.

Opening the Notebook Tabs Editor:

Select the TcsNotebook component on the form and then use one of the following techniques:

- press the secondary mouse button while the mouse pointer is positioned over the form and then choose the Edit Tabs command
- click the ellipsis button in the Value column for the Pages property
- double click in the Value column for the Pages property
- press Ctrl+Enter after moving to the Value column for the Pages property

Edit Button

Choose the Edit button to edit the Caption, Help Context, Enabled or Visible status of an existing tab. The Edit Tab dialog box will be displayed.

Add Button

Choose the Add button to add a new tab. The Add Tab dialog box will be displayed. New tabs are always added to the end of all existing tabs. If you want to insert a new tab you can add it and then use the Move button to change its position in the list of tabs.

Delete Button

Choose the Delete button to delete an existing tab. You will be asked to confirm that you want to delete the highlighted tab.

Up/Down Buttons

Choose the Up or Down buttons to change the position of an existing tab.

Choose OK or press Esc to close the editor.

Edit Tab Dialog Box

Applies to

TcsNotebook

Use the Edit Tab dialog box to edit the details for an existing tab in a TcsNotebook control. The Caption can contain an ampersand prefix to indicate the letter to be underlined.

Enabled

Remove the check from the Enabled check-box to prevent selection of the tab at run-time. You can use the TabPageEnabled method of TcsNotebook to enable/disable individual tabs at run-time.

Visible

Remove the check from the Visible check-box to prevent the tab from showing. You can use the TabPageVisible method of TcsNotebook to show/hide individual tabs at run-time.

Add Tab Dialog Box

Applies to

TcsNotebook

Use the Add Tab dialog box to add the details for a new tab in a TcsNotebook control. Refer to the Edit Tab Dialog Box for more information.

TcsPage Component

See also Properties

A TcsPage component is used for each page of a TcsNotebook component.

See Also

Pages

TabBitmap

TabCaption

TabNumGlyphs

TabPageEnabled

TabPageIndex

TabPageVisible

Properties

Bitmap

NumGlyphs

PageEnabled

PageVisible

TabBitmap Property

Applies to

TcsNotebook component

Declaration

property TabBitmap[Index: Integer]: TBitmap;

Description

Run-time only. The TabBitmap property gives you access to the bitmap for each tab. The Index value corresponds to the tab's page number. TabBitmap will return **nil** if the Index value is out of range.

Note that:

```
csNotebook1.TabBitmap[0].LoadFromFile('map.bmp');
```

is equivalent to:

```
TcsPages(csNotebook1.Pages.Objects[0]).Bitmap.LoadFromFile('map.bmp');
```

TabCaption Property

Applies to

TcsNotebook component

Declaration

property TabCaption[Index: Integer]: **String**;

Description

Run-time only. The TabCaption property gives you access to the caption text for each tab. The Index value corresponds to the tab's page number. TabCaption will return an empty string if the Index value is out of range.

Note that:

```
csNotebook1.TabCaption[0] := 'Tab 0';
```

is equivalent to:

```
TcsPages(csNotebook1.Pages.Objects[0]).Caption := 'Tab 0';
```

TabNumGlyphs Property

Applies to

TcsNotebook component

Declaration

property TabNumGlyphs[Index: Integer]: TTabNumGlyphs;

Description

Run-time only. The TabNumGlyphs property gives you access to the NumGlyphs property for the specified tab. The Index value corresponds to the tab's page number. TabNumGlyphs will return 1 if the Index value is out of range.

TabPageIndex Property

Applies to

TcsNotebook component

Declaration

property TabPageIndex[**const** TabIdentity: **String**]: Integer;

Description

Run-time only. The read-only TabPageIndex property allows you to determine the page number of the tab with the specified identity. TabPageIndex will return -1 if no tab exists with the specified identity.

TabPageEnabled Property

Applies to

TcsNotebook component

Declaration

property TabPageEnabled[Index: Integer]: Boolean;

Description

Run-time only. The TabPageEnabled property gives you access to the enabled status for each tab. The Index value corresponds to the tab's page number.

TabPageVisible Property

Applies to

TcsNotebook component

Declaration

property TabPageVisible[Index: Integer]: Boolean;

Description

Run-time only. The TabPageVisible property gives you access to the visible status for each tab. The Index value corresponds to the tab's page number.

PageEnabled Property

Applies to

TcsPage

Declaration

property PageEnabled: Boolean;

Description

The PageEnabled property indicates if you can select the page at run-time. Note that each page also has an (inherited) Enabled property that will be True if the page is the current page and False otherwise.

PageVisible Property

Applies to

TcsPage

Declaration

property PageVisible: Boolean;

Description

The PageVisible property indicates if you can see the page at run-time. Note that each page also has an (inherited) Visible property that will be True if the page is the current page and False otherwise.

TcsProperEdit Component

Unit

CSProper

Description

The TcsProperEdit and TcsDBProperEdit components allow proper-case text input and editing. During input each letter which occurs after a delimiter (such as space, comma, hyphen, apostrophe etc.) is converted to upper-case and all other letters are converted to lower-case. The TcsProperEdit and TcsDBProperEdit components both have the same additional properties and events beyond their ancestor's classes (TEdit and TDBEdit respectively), they do however differ in their implementations.

In addition to the following properties and events, TcsProperEdit components have the same properties and events as TEdit components.

In addition to the following properties and events, TcsDBProperEdit components have the same properties and events as TDBEdit components.

Properties

ProperCase

Events

OnConvert

ProperCase Property

Applies to

TcsProperEdit, TcsDBProperEdit components

Declaration

property ProperCase: Boolean;

Description

The ProperCase property determines whether the text should be converted to proper-case. The default value is True. Set ProperCase to False to disable text conversion.

OnConvert Event

Applies to

TcsProperEdit, TcsDBProperEdit components

Declaration

property OnConvert: TcsConvertEvent;

Description

The OnConvert event occurs prior to the default conversion of the input text whenever the text is changed. Writing an event handler for this event allows you to pre-process the text before the default handling is performed and/or to block the default handling altogether.

TcsDBProperEdit Component

Unit

CSProper

Description

The TcsDBProperEdit has the same additional properties as TcsProperEdit but is descended from TDBEdit.

TcsConvertEvent Type

Unit

CSProper

Declaration

TcsConvertEvent = **procedure**(**var** AString: **String**; **var** Handled: Boolean) **of object**;

Description

The TcsConvertEvent type points to a method that is called when the specified text needs converting to proper-case. The assigned method can set Handled to True to indicate that it has handled the conversion and no further conversion is necessary.

CSProper Unit

The CSProper unit contains the declaration for the TcsProperEdit and TcsDBProperEdit components and their associated objects.

The following items are declared in the CSProper unit:

Components

TcsProperEdit

TcsDBProperEdit

Types

TcsConvertEvent

TcsHiResTimer Component

Properties Events

Unit

CSHRTime

Description

The TcsHiResTimer component is similar to the TTimer component but differs in that you can set timer intervals less than the minimum 55ms (18.2 times per second) allowed with a TTimer. TcsHiResTimer allows an Interval as small as 1 millsecond. The timer events generated by TcsHiResTimer also have a higher priority than those generated by a TTimer thus ensuring that they are more likely to be processed when needed. The WM_TIMER messages used by TTimer are generally ignored by Windows while there are other messages pending and multiple pending WM_TIMER messages will actually be combined into a single message.

Important Note:

The 16-bit (Delphi 1) TcsHiResTimer component depends on the file CSTIME16.DLL. You should make sure that this DLL is available before running an application (16-bit) using the component. An application will still run if the DLL is not found but no timer events will occur. Windows will attempt to find the DLL by looking in the following locations (in this order):

1. The current directory.
2. The Windows directory (the directory containing WIN.COM).
3. The Windows system directory (the directory containing such system files as GDI.EXE).
4. The directory containing the executable file for the current task.
5. The directories listed in the PATH environment variable.
6. The list of directories mapped in a network.

The 32-bit (Delphi 2) TcsHiResTimer component doesn't require a DLL file.

Properties

DLLLoaded (run-time only)

Enabled

Interval

OneShot

Resolution

Events

OnTimer

DLLLoaded Property

Applies to

TcsHiResTimer component

Declaration

property DLLLoaded: Boolean;

Description

Delphi 1 (16-bit)	The DLLLoaded property indicates if the DLL (Dynamic Link Library) required by the TcsHiResTimer component was found and loaded successfully. If DLLLoaded is False no timer events will occur. The name of the required DLL file is CSTIME16.DLL.
Delphi 2 (32-bit)	DLLLoaded will always be True. No DLL is used.

Enabled Property

Applies to

TcsHiResTimer component

Declaration

property Enabled: Boolean;

Description

The Enabled property determines whether the component will generate OnTimer events. The default value is True.

Interval Property

Applies to

TcsHiResTimer component

Declaration

property Interval: UINT;

Description

The Interval property determines the interval in milliseconds (1ms = 1/1000th of a second) between successive timer events. The default value is 100ms. For example, to generate timer events 50 times per second you would set Interval to 20.

You should use the largest possible value appropriate for your needs. For example, if you only need to generate a timer event for an alarm in an appointment system alarm you could use an Interval of 1000 (1 second), using a smaller Interval would be wasteful of CPU time. System overhead, i.e. the amount of CPU time used, will increase as the Interval value is decreased.

The Resolution value can affect the accuracy of the Interval.

OneShot Property

Applies to

TcsHiResTimer component

Declaration

property OneShot: Boolean;

Description

The OneShot property determines whether Enabled will automatically be set to False after the OnTimer event occurs. This can be used to reduce system overhead if you need to generate timer events of varying, but known, lengths. The default value is False.

For example, if you have some MIDI data to output you can send the first note to be played and set the Interval so that the next timer event occurs when the next note needs to be played. This is more efficient than setting the Interval to a small value and then continually checking (in each OnTimer event) if the next note needs to be played yet.

Resolution Property

Applies to

TcsHiResTimer component

Declaration

property Resolution: UINT;

Description

The Resolution property determines the accuracy of the timer events in milliseconds. The default value is 100ms. In general you can usually use a Resolution value equal to the Interval value. Using Resolution values smaller than necessary will just increase system overhead, i.e. the amount of CPU time used.

OnTimer Event

Applies to

TcsHiResTimer component

Declaration

property OnTimer: TNotifyEvent;

Description

The OnTimer event occurs at successive time intervals as specified by the Interval value. If OneShot is True only one event will occur whenever the component is Enabled. No OnTimer events will occur if Enabled is False. Unless using OneShot timer events, the code you place in the OnTimer event handler should be able to execute in as little time as possible.

If your OnTimer event handler takes too long to execute it is possible (depending on the Interval value being used) that another (and another and...) timer event could occur before the previous timer event has been handled. Your program would continually be handling the timer events and would probably hang were this situation (time needed to execute OnTimer event handler > Interval) to continue.

CSHRTime Unit

The CSHRTime unit contains the declaration for the TcsHiResTimer component.

The following items are declared in the CSHRTime unit:

Components

TcsHiResTimer

TcsRankListBox Component

Properties

Unit

CSRankLB

Description

The TcsRankListBox component is a TListBox descendant in which the order of items can be changed at run-time by dragging them with the mouse (or by using the keyboard). Two modes of moving items using the mouse are possible, one where the item is moved as it is dragged (MoveOnDrag = True), the other where the item is only moved when the mouse button is released (MoveOnDrag = False).

Moving an item using the mouse:

Use Shift+Left-Click to select the item and then drag the item (keeping the Shift key and left mouse button depressed) to its new position. Release the mouse button and Shift key when the item is in the desired position.

Moving an item using the keyboard:

Select the item to be moved. Hold down the Shift key while using the Up, Down, Home or End keys to move the item to its new position.

In addition to the new MoveOnDrag property a TcsRankListBox component inherits all the properties and events of TListBox. However, the following properties have been made read-only and can't be changed:

Columns	*Always 0 (zero)*
DragCursor	*Always crDrag*
DragMode	*Always dmManual*
ExtendedSelect	*Always False*
MultiSelect	*Always False*
Sorted	*Always False*

Properties

MoveOnDrag

MoveOnDrag Property

Applies to

TcsRankListBox component

Declaration

property MoveOnDrag: Boolean;

Description

The MoveOnDrag property determines whether the item's position will be changed immediately (MoveOnDrag = True) as it is being dragged with the mouse or only when it is dropped (MoveOnDrag = False). The default value is True. When the keyboard is used to move an item the position will be changed immediately, regardless of the MoveOnDrag setting.

CSRankLB Unit

The CSRankLB unit contains the declaration for the TcsRankListBox component.

The following items are declared in the CSRankLB unit:

Components

TcsRankListBox

TcsAutoDefaults Component

Properties

Unit

CSADMain

Description

The TcsAutoDefaults component allows you to automatically apply previously stored default property values to new components dropped onto a form at design-time. You can also apply default property values to existing components on a form by using the Edit Automatic Defaults dialog box.

The component works by allowing you to store 'default' components in a file, called an AutoDefaults File. Default property values apply on a component type (class) basis (including descendant classes). An AutoDefaults file can contain any number of default components. The size of the AutoDefaults file will depend on the type and number of default components it contains.

Once you have defined your AutoDefaults file it is simply a matter of adding a TcsAutoDefaults component to a form and then whenever a new component is added to the form the component's properties will be changed to that of the 'default' component.

TcsAutoDefaults components on different forms can share the same AutoDefaults file, i.e. you will usually only need one file though you can have separate files for different types of forms or different projects.

TcsAutoDefaults components are ignored at run-time.

Using TcsAutoDefaults

To use the TcsAutoDefaults component you need to:

1. Add the component to a form.
2. Change the Filename property to the desired name.
3. Add a new component, e.g. a Label component, and set its properties to the values you want to be the default values for a TLabel. You should only change those properties which you would always change (to the same value) for every new Label.
4. Double-click the TcsAutoDefaults component or select it and then right-click and choose Edit. The Edit Automatic Defaults dialog box will be displayed. It is important to note that this step is only necessary in defining the 'default' component and is not necessary each time you want to later auto-default new components added to forms.
5. Select the label component you just edited from the Component/Class list and then choose the Add button. That component has now been saved as the 'default' Label component in the AutoDefaults file. The "Classes with defaults:" list will now show TLabel.
6. Close the Automatic Property Defaults form by pressing Esc or choosing the Close button.
7. Now add a new Label component to your form, you will see that its properties are changed to those of the 'default' component. Note that some properties (such as Left, Top) are not defaulted. The Name property is never defaulted.

If you want to change the defaults for a particular component you just need to repeat steps 3 to 5.

Setup

Prior to using this component you need to add the following information to your DELPHI.INI file. Please note that the UnsupportedClasses list should all be on one line (up to and including TSpinEdit), not on

two lines as shown below:

```
[ClassicSoftware.AutoDefaults]
DefaultFilename=c:\delphi\noname.adf
Extension=adf
UnsupportedClasses=TMainMenu;TPopupMenu;TTabSet;TDBNavigator;
                   TDBLookupCombo;TSpinButton;TSpinEdit
Ignore0=TControl.Left
Ignore1=TControl.Top
Ignore2=TComponent.Caption
Ignore3=TComponent.Text
Ignore4=TRadioGroup.Items
Ignore5=TDBRadioGroup.Items
Ignore6=TTabbedNotebook.Pages
Ignore7=TcsNotebook.Pages
```

Description of INI file settings

DefaultFilename specifies the value to be used for the Filename property when a new TcsAutoDefaults component is added to a form (thus saving you from having to type the filename every time). Change the setting to a suitable name.

Extension is the default filename extension to be used for AutoDefaults files when no extension is specified.

UnsupportedClasses are those classes within the standard Delphi environment which cannot be saved as default components, there may be others in addition to those shown above which you come across from 3rd party component libraries. Attempting to save a component which is an unsupported class should not cause any problems and you can later add it to the UnsupportedClasses section if you get an 'Ignore this component' message when loading the AutoDefaults file.

Ignore<n> settings specify which properties of a default component are to be ignored because they are not relevant or cause problems. For example, when you drop a new component on a form you want it to stay where you dropped it, thus the Top and Left properties should not be set to the default values. The Ignore<n> settings in DELPHI.INI are used as the initial Ignore values when an AutoDefaults file is first loaded. If the AutoDefaults file contains alternate values which have been specified from within the TcsAutoDefaults component editor, they will be used instead.

Properties

Active

Filename

ShowHints

Active Property

Applies to

TcsAutoDefaults component

Declaration

property Active: Boolean;

Description

The Active property determines whether the TcsAutoDefaults component will respond to the adding of new components to the form. The default value is True. When a new component is added to a form and Active is True, any existing default property values for that type of component will be applied. When Active is False, no action is taken when new components are added to the form.

Filename Property

Applies to

TcsAutoDefaults component

Declaration

property Filename: **String**;

Description

The Filename property determines the name of the AutoDefaults file in which the 'default' components will be stored. The DefaultFilename setting in the [ClassicSoftware.AutoDefaults] section of your DELPHI.INI file is used as the default value. If you specify the name of an existing AutoDefaults file the defaults in that file will be loaded. If you specify the name of a file which is not an AutoDefaults file you will get an error message. If you omit the filename extension the Extension setting in the [ClassicSoftware.AutoDefaults] section of your DELPHI.INI file will be used. AutoDefaults filenames can use any extension but the recommended extension is "ADF".

ShowHints Property

Applies to

TcsAutoDefaults component

Declaration

property ShowHints: Boolean;

Description

The ShowHints property determines whether hints will be shown for buttons on the component editor (Edit Automatic Defaults dialog box) at design-time. You can show the component editor by double clicking the component or by right clicking the component and choosing the Edit command.

CSADMain Unit

The CSADMain unit contains the declarations for the TcsAutoDefaults component.

The following items are declared in the CSADMain unit:

Components

TcsAutoDefaults

Exceptions

EcsADStreamError

Constants

WM_ApplyDefaults

Edit Automatic Defaults Dialog Box

Applies to

TcsAutoDefaults

The Edit Automatic Defaults dialog box shows all the components on the current form in alphabetical order. The class for each component is also shown. On the right hand side is a list which shows the classes which have previously stored defaults.

Add button

Press this button to add the property values of the highlighted component as the defaults for that component's class. The defaults are saved immediately in the AutoDefaults file. If the class of the highlighted component already has defaults then they will be replaced with the new defaults.

Set button

Press this button to set the property values of the highlighted component to the default values for that component's class. If the highlighted component's class has no defaults then no changes will be made to the component's properties.

Delete button

Press this button to delete the defaults for the classes highlighted in the "Classes with defaults" list. You can make multiple/extended selections in the list to delete the defaults for multiple classes.

Delete All button

Press this button to delete the defaults for all classes in the "Classes with defaults" list, ***whether they are highlighted or not***. You will be asked to confirm that you want to delete all defaults before this is done. Note that the only way of restoring the default values after deleting all defaults is by shutting down Delphi and then restoring the original AutoDefaults file from a backup copy (or by re-adding the defaults for each class from scratch).

Ignore button

Press this button to define the list of property values that should be ignored when applying defaults. Each item in the list should be in the format <class>.<property> where <class> is the class containing the property and <property> is the property name. For example, to ignore the Left property (because you don't want components added to a form to all be defaulted to the same position) of all TControl descendants you would add TControl.Left to the list. (This is in fact standard so you don't actually need to add TControl.Left in this case.) Each item in the list should be on a separate line. The specified properties will be ignored in the specified class and all its descendants. Thus including TControl.Left in the list actually causes TEdit.Left, TLabel.Left etc. to be ignored too because these are all TControl descendants.

Close button

Press this button or press the ESC key to close the dialog box. Changes are automatically saved so there is no difference between using the Close button and pressing ESC.

TcsFormPanel Component

Properties

Unit

CSFrmPnl

Description

The TcsFormPanel component is similar to a TPanel but adds two new properties -- Form (run-time only) and FormName -- which allow a form to be displayed on the panel's surface. Thus, a TcsFormPanel can be used as a sub-form component to allow one form to be placed on another form, on another panel, on a notebook page etc. This allows greater modularity of your code by allowing you to keep the code for the sub-form separate from its container (form/panel/notebook) and can also allow you to re-use the same sub-form on multiple forms.

Certain properties which are available in TPanel are not applicable in TcsFormPanel -- because the form will be occupying the whole of the TcsFormPanel's surface -- and have been removed. In some cases (DragCursor, Font, OnClick, OnDragDrop etc.) there are equivalent properties/events in the sub-form that you can use instead. The TPanel properties and events which are not also present in TcsFormPanel components are listed below:

Properties removed

Alignment, DragCursor, DragMode, Ctl3D, Font, Locked, ParentCtl3D, ParentFont, ParentShowHint, PopupMenu, ShowHint.

Events removed

OnClick, OnDblClick, OnDragDrop, OnDragOver, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnResize.

Properties

Form (run-time only)

FormName

Form Property

Example

Applies to

TcsFormPanel component

Declaration

property Form: TForm;

Description

Run-time only. The Form property determines the current form to display on the panel. When using auto-created forms you can also use the FormName property. Note that when you assign a new form to the Form property the previous form is only hidden and not closed or released.

It is your responsibility to create (before assigning to the Form property) and destroy (after hiding) any forms which are not auto-created by the application. The example shows how to do this within another form. The source code for CSFPMAIN.PAS in the CSFPDEMO project illustrates how to dynamically create and destroy forms as each page on a TcsNotebook is displayed.

FormName Property

Applies to

TcsFormPanel component

Declaration

property FormName: **String**;

Description

The FormName property can be used to indicate the name of the auto-created form to be displayed on the panel. You can determine all forms which are being auto-created by choosing **Options | Project** from the Delphi menu and then choosing the **Forms** tab. You should not use the FormName property for a form that is not being auto-created -- use the Form property instead.

CSFrmPnl Unit

The CSFrmPnl unit contains the declaration for the TcsFormPanel component.

The following items are declared in the CSFrmPnl unit:

Components

TcsFormPanel

Example

The following example shows how you would create a sub-form (Form2) in Form1's OnCreate event handler. Form1 is specified as the owner of the sub-form so that it will take care of destroying the sub-form. Remember to include the name of the unit containing the sub-form in the **uses** clause of the unit which is creating the sub-form, i.e. for the following example you would include Unit2 in the **uses** clause of Unit1 (this assumes that Unit1 contains the definition for TForm1 and Unit2 contains the definition for TForm2):

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  csFormPanel1.Form := TForm2.Create(Self);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if (csFormPanel1.Form <> nil) then
    csFormPanel1.Form.Close;
end;
```

TcsComboBox Component

Unit

CSXCombo

Description

The TcsComboBox component is functionally the same as the TComboBox component except that it will correctly remember the current item when changing pages on a TcsNotebook component which has the SaveResources property set to True. All properties and events of TcsComboBox are the same as for TComboBox, only the behaviour is changed. The TcsComboBox component can also be used instead of TComboBox on other 'paged' components such as TNotebook and TTabbedNotebook when other resource saving mechanisms (via direct calls to DestroyHandle) are being used.

TcsDBComboBox Component

Unit

CSXCombo

Description

The TcsDBComboBox component is functionally the same as the TDBComboBox component except that it will correctly remember the current item when changing pages on a TcsNotebook component which has the SaveResources property set to True. All properties and events of TcsDBComboBox are the same as for TDBComboBox, only the behaviour is changed. The TcsDBComboBox component can also be used instead of TDBComboBox on other 'paged' components such as TNotebook and TTabbedNotebook when other resource saving mechanisms (via direct calls to DestroyHandle) are being used.

CSXCombo Unit

The CSXCombo unit contains the declarations for alternative ComboBox components which can be used on the pages of TcsNotebook components.

The following items are declared in the CSXCombo unit:

Components

TcsComboBox

TcsDBComboBox

TcsGrid Component

Properties Methods

Unit

CSGrid

Description

The TcsGrid component is similar to the TStringGrid component but differs in that it uses a TcsStringTable to store the cells' data rather than a sparse-array as used by TStringGrid. The main ramification of this is that each cell already contains a zero length string and insertion and deletion of columns and rows can be performed quickly using the InsertColumn, DeleteColumn, InsertRow and DeleteRow methods. Additional changes allow the grid to be used as though it were a multiple-selection columned ListBox by using the ExtendedSelect and MultiSelect properties (in conjunction with the goRowSelect setting in Options).

A current row indicator will be drawn in the last fixed column when MultiSelect and goRowSelect have both been set to True and you have one or more fixed columns.

Properties

TcsGrid inherits all the properties of TDrawGrid. The following properties have been added or changed:

Cells (run-time only)
Data (run-time only)
DataCells (run-time only)
DataObjects (run-time only)
ExtendedSelect
MultiSelect
Objects (run-time only)
Selected (run-time only)

Methods

TcsGrid inherits all the methods of TDrawGrid. The following methods have been added or changed:

AddColumn
AddRow
ClearDataCells
ClearSelected
DeleteColumn
DeleteRow
InsertColumn
InsertRow
InvalidateRow

Cells Property

Applies to

TStringGrid, TcsGrid components; TcsStringTable object

Declaration

property Cells[ACol, ARow: Integer]: **String**;

Description

Run-time only. The Cells property allows access to each individual cell of the grid. ACol specifies the column coordinate and ARow specifies the row coordinate. Column coordinates range from 0 to ColCount - 1. Row coordinates range from 0 to RowCount - 1. To reference cells which are not part of the fixed rows or columns in a TcsGrid you can use the DataCells property, i.e. DataCells[0, 0] is equivalent to Cells[FixedCols, FixedRows].

Unlike TStringGrid, with TcsGrid you do not have to store something in a cell before using it, all cells are initialised to a zero length string.

Data Property

Applies to

TOutlineNote, TcsGrid components

Declaration

property Data: TcsStringTable;

Description

Run-time only. The Data property allows access to the data structure which holds the data for the cells of the grid.

DataCells Property

Applies to

TcsGrid component

Declaration

property DataCells[ACol, ARow: Integer]: **String**;

Description

Run-time only. The DataCells property allows access to each individual 'data' cell of the grid. A 'data' cell is any of the non-fixed cells. ACol specifies the column coordinate and ARow specifies the row coordinate. Column coordinates range from 0 to ColCount -FixedCols - 1. Row coordinates range from 0 to RowCount - FixedRows - 1. DataCells[0, 0] is equivalent to Cells[FixedCols, FixedRows].

DataObjects Property

Applies to

TcsGrid component

Declaration

property DataObjects[ACol, ARow: Integer]: TObject;

Description

Run-time only. The DataCells property allows access to the objects associated with each individual 'data' cell of the grid. A 'data' cell is any of the non-fixed cells. ACol specifies the column coordinate and ARow specifies the row coordinate. Column coordinates range from 0 to ColCount -FixedCols - 1. Row coordinates range from 0 to RowCount - FixedRows - 1. DataCells[0, 0] is equivalent to Cells[FixedCols, FixedRows].

Objects Property

Applies to

TStringList, TStrings objects; TStringGrid, TcsGrid components

Declaration

property Objects[ACol, ARow: Integer]: TObject;

Description

Run-time only. The Objects property allows access to the object associated with each individual cell of the grid. ACol specifies the column coordinate and ARow specifies the row coordinate. Column coordinates range from 0 to ColCount - 1. Row coordinates range from 0 to RowCount - 1. To reference objects associated with cells in a TcsGrid which are not part of the fixed rows or columns you can use the DataObjects property, i.e. DataObjects[0, 0] is equivalent to Objects[FixedCols, FixedRows].

Selected Property

Applies to

TDBListBox, TDirectoryListBox, TFileListBox, TListBox, TcsGrid components; TcsStringTable object

Declaration

property Selected[ARow: Integer]: Boolean;

Description

Run-time only. The Selected property indicates if the specified row has been selected.

ExtendedSelect Property

Applies to

TListBox, TcsGrid components

Declaration

property ExtendedSelect: Boolean;

Description

The ExtendedSelect property determines if the user can select a range of rows in the grid. It is only effective when MultiSelect is True and goRowSelect has been selected in Options.

MultiSelect Property

Applies to

TListBox, TFileListBox, TcsGrid components

Declaration

property MultiSelect: Boolean;

Description

The MultiSelect property determines if the user can select more than one item in the grid. This property is only effective when goRowSelect has been selected in Options. A current row indicator will be drawn in the last fixed column (when you have one or more fixed columns) when MultiSelect and goRowSelect are set to True.

AddColumn Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure AddColumn;

Description

The AddColumn method adds a new (last) column. The new column will have the same number of rows as existing columns and each cell will contain a zero length string.

AddRow Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure AddRow;

Description

The AddRow method adds a new (last) row. The new row will have the same number of columns as existing rows and each cell will contain a zero length string.

ClearDataCells Method

Applies to

TcsGrid component

Declaration

procedure ClearDataCells;

Description

The ClearDataCells method sets the contents of each 'data' cell (all non-fixed cells) to a zero length string and clears any selected rows.

ClearSelected Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure ClearSelected;

Description

The ClearSelected method sets the Selected status for all rows to False.

DeleteColumn Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure DeleteColumn(ACol: Integer);

Description

The DeleteColumn method deletes the specified column. If there are objects associated with the cells of the column to be deleted you should free the objects before using DeleteColumn.

DeleteRow Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure DeleteRow(ARow: Integer);

Description

The DeleteRow method deletes the specified row. If there are objects associated with the cells of the row to be deleted you should free the objects before using DeleteRow.

InsertColumn Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure InsertColumn(ACol: Integer);

Description

The InsertColumn method inserts a new column at the specified column position.

InsertRow Method

Applies to

TcsGrid component; TcsStringTable object

Declaration

procedure InsertRow(ARow: Integer);

Description

The InsertRow method inserts a new row at the specified row position.

InvalidateRow Method

Applies to

TcsGrid component

Declaration

procedure InvalidateRow(ARow: Integer);

Description

The InvalidateRow method invalidates the whole of the specified row, including fixed cells and any partially visible cells on the right hand side of the grid.

CSGrid Unit

The CSGrid unit contains the declaration for TcsGrid.

The following items are declared in the CSGrid unit:

Components

TcsGrid

TcsStringTable Object

Properties Methods

Unit

CSStrTbl

Description

The TcsStringTable class defines a dynamic two dimensional string array which is used as the storage class for TcsGrid cells but which can also be used for other purposes.

Properties

Cells (run-time only)
ColCount (run-time only)
Objects (run-time only)
RowCount (run-time only)
Selected (run-time only)
SelectedRows (run-time only)

Methods

AddColumn
AddRow
ChangeSize
Clear
ClearSelected
Create
DeleteColumn
DeleteRow
InsertColumn
InsertRow
MoveColumn
MoveRow

ColCount Property

Applies to

TDrawGrid, TStringGrid, TcsGrid components; TcsStringTable object

Declaration

property ColCount: Integer;

Description

Run-time only. The ColCount property indicates the number of columns in the grid/table. ColCount must be greater than zero.

RowCount Property

Applies to

TDrawGrid, TStringGrid, TcsGrid components; TcsStringTable object

Declaration

property RowCount: Integer;

Description

Run-time only. The RowCount property indicates the number of rows in the grid/table. RowCount must be greater than zero.

SelectedRows Property

Applies to

TcsStringTable object

Declaration

property SelectedRows: TStringList;

Description

Run-time and read only. The SelectedRows property contains details of the selected rows. Each item in the list is the IntToStr() value of the number of the row which was selected. The list of selected rows is adjusted whenever rows are inserted, deleted or moved.

ChangeSize Method

Applies to

TcsStringTable object

Declaration

procedure ChangeSize(NewColCount, NewRowCount: LongInt);

Description

The ChangeSize method allows the number of columns and rows in the table to be changed. You can use ChangeSize instead of making multiple calls to AddColumn, DeleteColumn, AddRow or DeleteRow.

Clear Method

Applies to

TcsStringTable object

Declaration

procedure Clear;

Description

The Clear method allows the contents of all cells to be set to a zero length string. If you have objects associated with each cell these will be unaffected by this method.

Create Method

Applies to

TcsStringTable object

Declaration

constructor Create(NumCols, NumRows: Integer);

Description

The Create method creates a new TcsStringTable object with the specified number of columns and rows. The minimum size table that can be created is a 1 x 1 size table.

MoveColumn Method

Applies to

TcsStringTable object

Declaration

procedure MoveColumn(FromIndex, ToIndex: LongInt);

Description

The MoveColumn method moves the column at position FromIndex to position ToIndex.

MoveRow Method

Applies to

TcsStringTable object

Declaration

procedure MoveRow(FromIndex, ToIndex: LongInt);

Description

The MoveRow method moves the row at position FromIndex to position ToIndex.

CSStrTbl Unit

The CSStrTbl unit contains the declaration for TcsStringTable.

The following items are declared in the CSStrTbl unit:

Types

TcsStringTable

TabAtPos Method

Applies to

TcsNotebook component

Declaration

function TabAtPos(X, Y: Integer): Integer;

Description

The TabAtPos method returns the index position (0..Pages.Count - 1) of the tab which contains the specified point. If no tab containing the specified point is found then -1 is returned.

TcsSculptButton Component

Properties Methods Events

Unit

CSSclptB

Description

The TcsSculptButton component displays a button of the shape, color and texture specified by its Bitmap property. Areas of the bitmap which are the transparent color will not be shown, allowing the background underneath the button to show through. A 3D bevel can be added automatically by specifying the BevelWidth. Areas of the button can also be Speckled. The TcsSculptButton component is a non-windowed component that is a descendant of TGraphicControl.

In addition to these properties, methods and events, this component also has the properties and methods that apply to all controls.

Properties

AutoSize
BevelHighlightColor
BevelShadowColor
BevelWidth
Bitmap
BitmapDown (run-time only)
BitmapUp (run-time only)
BorderColor
PreciseClick
PreciseShowHint
Speckled
SpeckleOpaqueColor
SpeckleTransparentColor
TextPosition
TextX
TextY

Methods

PtInMask

Events

OnClick	OnMouseDown
OnDragDrop	OnMouseMove
OnDragOver	OnMouseUp
OnEndDrag	

CSSclptB Unit

The CSSclptB unit contains the declaration for the TcsSculptButton component.

The following items are declared in the CSNoteBk unit:

Components

TcsSculptButton

Types

TcsBevelWidth = 0..2;

TcsTextPosition = (tpCentered, tpXY);

PtInMask Method

Applies to

TcsSculptButton component

Declaration

function PtInMask(**const** X, Y: Integer): Boolean; **virtual**;

Description

The PtInMask method returns True if the specified point is inside the mask for the button, i.e. that the point is in the non-transparent part of the bitmap.

AutoSize Property

Applies to

TcsSculptButton, TDBEdit, TDBText, TEdit, TImage, TLabel, TMaskEdit, TOLEContainer components.

Declaration

property AutoSize: Boolean;

Description

The AutoSize property determines whether the component automatically resizes to match the size of the bitmap assigned to the Bitmap property. The default value is True.

BevelHightlightColor Property

Applies to

TcsSculptButton

Declaration

property BevelHightlightColor: TColor;

Description

The BevelHightlightColor property determines the color used for the highlighted part of the 3D bevel effect applied to the bitmap's outline. The default value is clBtnHighlight.

BevelShadowColor Property

Applies to

TcsSculptButton

Declaration

property BevelShadowColor: TColor;

Description

The BevelShadowColor property determines the color used for the shadowed part of the 3D bevel effect applied to the bitmap's outline. The default value is clBtnShadow.

BevelWidth Property

Applies to

TcsSculptButton, TPanel components.

Declaration

property BevelWidth: TcsBevelWidth;

Description

The BevelWidth property determines the width of the 3D bevel effect applied to the bitmap's outline. The default (and maximum) value is 2.

Bitmap Property

Applies to

TcsSculptButton, TcsNotebook components

Declaration

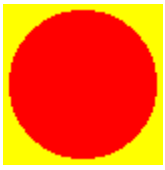
property Bitmap: TBitmap;

Description

The Bitmap property of TcsSculptButton components determines the shape, color and texture of the button. Areas of the bitmap which are the transparent color will not be shown, allowing the background to be seen. If using an automatic 3D bevel effect (BevelWidth > 0) then bitmaps with intricate outlines may not produce the desired effect -- using a narrower bevel may help. The bitmap used should also have a border (of the transparent color) of at least 3 pixels wide to prevent clipping from occurring when the 3D bevel effect is added.

Examples:

The following simple bitmap would produce a round red button. The yellow areas would become transparent and wouldn't be seen:



The following bitmap would produce an oval red button with a blue edge. The yellow areas would become transparent. If you then set Speckled to True and SpeckleTransparentColor to clRed the red area would become speckled -- pixels would alternate between being transparent (to let the color of the pixel in the component underneath the button to show through) and being the color specified by the SpeckleOpaqueColor property.

Transparent areas aren't limited to just the outside edges of the bitmap. The following bitmap would produce a red donut-shaped button. All yellow areas would become transparent:

BitmapDown Property

Applies to

TcsSculptButton

Declaration

property BitmapDown: TBitmap;

Description

Run-time only. The read-only BitmapDown property allows access to the bitmap used for the 'down' state of the button.

BitmapUp Property

Applies to

TcsSculptButton

Declaration

property BitmapUp: TBitmap;

Description

Run-time only. The read-only BitmapUp property allows access to the bitmap used for the 'up' state of the button.

BorderColor Property

Applies to

TcsSculptButton

Declaration

property BorderColor: TColor;

Description

The BorderColor property determines the color of the border around the outline of the button. The default value is clBlack. If you don't want a border then you can set BorderColor to the transparent color color of the Bitmap.

PreciseClick Property

Applies to

TcsSculptButton

Declaration

property PreciseClick: Boolean;

Description

The PreciseClick property determines how the button will respond to mouse clicks and mouse pointer (cursor) movement. The default value is True. When PreciseClick is True any mouse clicks in the transparent areas of the button will be ignored and the mouse pointer will only change (if the Cursor property is other than crDefault) when over non-transparent areas of the button. When PreciseClick is False, the button is treated as though it were a regular rectangular shaped button. You may need to set PreciseClick to False if for example your users are confused when nothing happens when they click 'on' a button but over a transparent area (imagine a donut shape and they click in the hole) or if children use your program and you want to allow them to click on or near the button (within its rectangle).

PreciseShowHint Property

Applies to

TcsSculptButton

Declaration

property PreciseShowHint: Boolean;

Description

The PreciseClick property determines whether hints will only be shown when the mouse pointer (cursor) is positioned over a non-transparent part of the button. The default value is True.

It is important to note that the effect of PreciseShowHint being True may not be quite what you expect -- because the Application may still display the hint for the parent (if the parent has ShowHint = True). Consider the situation where a TcsSculptButton has been placed over a TImage and the button, image and form all have ShowHint set to True. In this case PreciseShowHint will result in the hint for the form being shown (because it is the parent of the button) rather than the hint for the image when the cursor is positioned over the transparent parts of the button.

Speckled Property

Applies to

TcsSculptButton

Declaration

property Speckled: Boolean;

Description

The Speckled property determines whether the surface of the button will be speckled (alternating pixels either transparent or the color specified by the SpeckleOpaqueColor property). Speckling will occur on areas of the button which match the value of the SpeckleTransparentColor property. The default value is False.

SpeckleOpaqueColor Property

Applies to

TcsSculptButton

Declaration

property SpeckleOpaqueColor: TColor;

Description

The SpeckleOpaqueColor is used when Speckled is True and is applied (in a checkerboard pattern) to the areas of the button's Bitmap which are the same color as SpeckleTransparentColor. The default value is clWhite.

SpeckleTransparentColor Property

Applies to

TcsSculptButton

Declaration

property SpeckleTransparentColor: TColor;

Description

The SpeckleTransparentColor is used when Speckled is True to determine the areas of the button's Bitmap which are to be speckled. The default value is clBlack. The speckles will be the color specified by the SpeckleOpaqueColor property.

TextPosition Property

Applies to

TcsSculptButton

Declaration

property TextPosition: TcsTextPosition;

Description

The TextPosition property determines where the button's Caption text will be displayed. The default value is tpCentered which will result in the caption being centered both horizontally and vertically within the button's rectangle. If you need to explicitly position the text, for example when using an irregularly shaped button, then set TextPosition to tpXY and specify the location using the TextX and TextY properties.

TextX Property

Applies to

TcsSculptButton

Declaration

property TextX: Integer;

Description

The TextX property is only used when TextPosition has been set to tpXY and is used along with TextY to determine the location where the button's Caption text will be displayed. TextX specifies the horizontal coordinate relative to the left side of the button's client rectangle. The default value is 0.

TextY Property

Applies to

TcsSculptButton

Declaration

property TextY: Integer;

Description

The TextY property is only used when TextPosition has been set to tpXY and is used along with TextX to determine the location where the button's Caption text will be displayed. TextY specifies the vertical coordinate relative to the top of the button's client rectangle. The default value is 0.

The transparent color for a bitmap is based on the color of the pixel at the bottom-left corner of the bitmap (or clWhite for monochrome bitmaps) and can be accessed using the TransparentColor property of TBitmap objects.