Contents

Introduction

Welcome to the world of cwComponents designed to enhance your productivity and add the Professional Touch to your applications.

This suite of 3 components delivers instant Cut Copy and Paste support not just for Edit fields, but also for **Grids** and **dbImage** fields. Without these components, this support would take a significant programming effort.

Overview

This component set was born because of the difficulty of coding generic cut copy and paste routines that work with dbGrids stringGrids and even dbImages. Especially in Delphi 1.0. It also fixes the bug in Delphi 1.0 dbImages that causes a GPF if you try to cut an image to the clipboard when the Stretch property is true.

Components

Other Information

Technical Support

Compuserve 73163,2765
From the Internet 73163.2765@compuserve.com

Additional Files

TBAR.RES contains the bitMaps for the 3 buttons.
TBAR1.DCR contains the icons for the components.

Known Problems

Does not support all 3 rd Party Grids. Contact us for specific information and additional modules with alternate code.

Registration

Registration and Pricing

Pricing (Subject to change without notice)

Without source	$19.00
With Source	$29.00

Via Compuserve [SWREG]

Go SWREG and follow the instructions.

Without source	SWREG 10762
With Source	SWREG 10763

We also take

VISA
Master Card
American Express

To order by Credit Card you must provide
Card Number, Expiration Date and **Name** as it appears on the card.
Please also include your Address and Telephone number.

Concerned about sending your card number over the Internet?
Fax it to us at 619-566-0210 USA

All Components are delivered by eMail.
INTERNET : Delivery by UUE encoded Zip file.

TcwClpBoardBar

This is a descendant of TcwToolBar

It is a TcwToolBar with three cwClpBoardButtons encapsulated as Cut Copy and Paste buttons.

Using this composite component will ensure a conforming appearance across all forms without needing precise GUI manipulation. It inherits all the neat features of the TcwToolBar and by increasing its width you can add additional buttons that will also be manipulated as in the TcwToolBar.

New Method

setEnabledState

This method controls the setEnabledState of all 3 encapsulated TcwClpBoardButtons.

Example

1. Create an appIdle method. If you already have one then just add the code to it.
2. Give cwClpBoardBar an informative name (not required)
3. Add the code shown below.
4. Point the Application.onIdle method to the new AppIdle method in the FormCreate event.

```
procedure TForm1.AppIdle(Sender: TObject; var Done: Boolean);
Begin
Try
 myClipBar.setEnabledState;
except
 on E:Exception do
  messageDlg(E.Message,mtError,[mbOK],0);
end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 Application.OnIdle := AppIdle;
end;
```

TcwClpBoardButton

The TcwClpBoardButton encapsulates all the necessary code for Cutting, Copying or Pasting within its click method. No code needed!

New property

Kind

.Values

bkCut	The default
bkCopy	
bkPaste	

Setting this property is all you have to do to turn it into the appropriate type of button including setting the correct
BEHAVIOUR, BITMAP and HINT.

New Method

setEnabledState

It has a 'setEnabledState' method which makes their Enabled state Data Aware.

Example

1. Create an appIdle method. If you already have one then just add the code to it.
2. Give each button an informative name (not required)
3. Add the code shown below.
4. Point the Application.onIdle method to the new AppIdle method in the FormCreate event.

```
procedure TForm1.AppIdle(Sender: TObject; var Done: Boolean);
Begin
Try
 butCut.setEnabledState;
 butCopy.setEnabledState;
 butPaste.setEnabledState;
except
 on E:Exception do
  messageDlg(E.Message,mtError,[mbOK],0);
end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 Application.OnIdle := AppIdle;
end;
```

TcwToolBar

This is a descendent ot TPanel that knows how to:-

Resize any components it contains when IT is resized.

Set its orientation to Horizontal or Vertical.

Dock itself to Top, Left, Top, Bottom or None.

New Properties

AutoSizeButtons
KeepSquare
Orientation
Dock

AutoSizeButtons

Values

True	
False	Default

Turns autoSizing On or Off. When On all contained buttons will be resized to exactly fill the toolbar.
You can also use this ability to make sure all components dropped onto the toolBar are the correct height and precisely aligned.
A great time saver!
Hint
By dropping bevel components onto the toolbar and setting their visible property to FALSE you can create spacers to seperate different logical groups of buttons.

KeepSquare

Values

True	
False	Default

Turn On or Off When On buttons will stay square when autoSized.
Can not be True if AutoSizeButtons is False.

Orientation

Values

toHorizontal	Default
toVertical.	

When the Orientation is changed the component knows how to reset any components it contains to a horizontal set or a vertical set.
WARNING : Components that are not approximately square may be truncated if orientation is changed. (e.g. Drop down combo)

Dock

Values

tdTop
tdLeft
tdBottom
tdRight

tdNone	Default

Dock itself to Top, Left, Right, Bottom or None.
It automatically takes care of reAligning any buttons it contains appropriately.

Tips

HINTS ON USE

TcwClpBoardButton

Use it as a non visual object (Visible = False) to easily enable Menu items with one line of code.

1. Give the button a descriptive name e.g. butCut
2. In the Menu event handler use butCut.Click;

If you set the menu item shortCut key to Ctrl-X you will get the added benefit of having the component code respond instead of windows native behaviour. This avoids the dbImage cutToClipboard GPF problem.

You can also handle Shift-Delete but that has to be done on each dbImage component in the KeyDown event.

Example

```
procedure   TForm1.DBImage1KeyDown(Sender: TObject; var Key: Word;   Shift: TShiftState);
begin
 if (shift = [ssShift]) and ( Key = VK_Delete) then begin
    Key := 0; {Prevent windows from seeing it}
    butCut.Click;
 end;
end;
```