Custom Clipboard Formats

by Xavier Pacheco

The Windows clipboard is a convenient resource that allows you to share information from your application with other Windows applications. Clipboard operations are especially handy when designing data entry screens – users can avoid having to re-enter data that has already been entered in a previous screen. By using the clipboard appropriately, you can reduce the work for your users and make your applications much more user-friendly.

Delphi quite handily provides the `TClipboard` class which encapsulates much of the clipboard functionality that you would otherwise have to code yourself. `TClipboard` already knows how to work with many formats and using `TClipboard` with these existing formats is explained quite well in Delphi's online help.

In this article, I'll show how to define your own custom formats that `TClipboard` doesn't yet know about. By defining a custom clipboard format, you can save data to the clipboard in any special format that you think would enhance your application, and make life easier for those using your application.

You might have seen Stefan Boether's example of how to store the text from a `TMemo` component to the clipboard in the Tips & Tricks column of Issue 2 of The Delphi Magazine. Stefan cleverly illustrates how to store the entire `TMemo`'s text using a `TMemoryStream` in conjunction with a `TClipboard` descendant.

Stefan created a `TClipboard` descendant class and added the special methods to allow the `TMemo` contents to be saved to and pasted from the clipboard. This approach required that you first free the original clipboard before instantiating the `TClipboard` descendant.

As an applications developer, Stefan's approach is ideal. Just be sure to free the clipboard only once in one unit. Additionally, place any other specialized clipboard routines in the same `TClipboard` descendant. In other words, only have one `TClipBoard` descendant. This is important because the `Clipboard` variable is global and you don't want multiple `TClipBoard` descendants freeing each other in each unit's initialization section.

If you are a component writer, you'll want to use the approach I present here. This approach makes your components aware of the clipboard, rather than the reverse. By using this approach, an applications developer using various components that are "clipboard aware" need not be concerned about `TClipboard` descendants freeing each other in the same application. Also, this is the approach that many of the Delphi components use.

Creating The Custom Clipboard Format

To illustrate creating a custom clipboard format, I've created the `TBirthDay` component, which is nothing more than a component wrapper around a record containing a person's name, age and birth date. The data is stored in the data types `string[100]`, `integer` and `TDateTime` respectively. Listing 1 (opposite) shows the source code for the BDAY.PAS unit.

As I said earlier, the `TBirthDay` component is just a component wrapper around a record with fields of different types. The main thing I want to illustrate here is how to copy data to the clipboard. In this example, the data consists of a person's name and birthday information.

You will notice that the `TBirthDay` class has a `TPersonRec` variable and two methods, `CopyToClipBoard` and `PasteFromClipBoard`. I've made the `TPersonRec` accessible through the property `Person`. Also notice the line in the unit's initialization section:

```
CF_BIRTHDAY :=
  RegisterClipBoardFormat(
  'CF_BIRTHDAY');
```

`RegisterClipBoardFormat` is a Windows API function that registers a special format with the Windows clipboard. This makes your special format available to applications that know how to work with this format. This statement also makes the format appear on `TClipBoard`'s list of formats, which you can access through `TClipBoard`'s `Formats` property. `RegisterClipBoardFormat` returns a value that indicates the newly registered format. Other applications that call this function and pass in the same string `CF_BIRTHDAY`, would receive the same value as when it was previously registered.

The `CopyToClipBoard` method is responsible for placing the data contained in `FPersonRec` onto the clipboard using `TClipBoard`'s `SetAsHandle` method. `SetAsHandle` takes the clipboard format variable and a `THandle` as parameters and places the data referenced by the `THandle` onto the clipboard in the specified format. In this example, I pass `CF_BIRTHDAY`, to indicate my custom format.

The code surrounding the call to `SetAsHandle` simply prepares a valid `THandle`. The line:

```
Data :=
  GlobalAlloc(GMEM_MOVEABLE,
  SizeOf(FPersonRec));
```

tells Windows to allocate `Sizeof(FPersonRec)` bytes on the global heap and to return a handle to that memory to the variable `Data`. A pointer to that memory area is retrieved with the line:

```
DataPtr := GlobalLock(Data);
```

The data is then moved to the memory block with the `Move()` procedure. In this method I copy

the data to the clipboard in two formats, CF_BIRTHDAY and CF_TEXT. In order to do this successfully, I must call the ClipBoard's method Open explicitly before I copy each format to the clipboard. Calling Open prevents subsequent calls to clipboard methods from erasing the clipboard's contents, thus allowing me to save multiple formats at once.

Note that I call GlobalFree on the variable Data only in the event of an exception. This is because once Data is copied to the clipboard, Windows is responsible for managing the memory occupied by Data, which includes freeing its memory.

TBirthday's PasteFromClipBoard performs the opposite of the CopyToClipBoard method. It retrieves the data in the CF_BIRTHDAY format using TClipboard's GetAsHandle method. This method takes the CF_BIRTHDAY value as a parameter and returns a handle to the requested data. PasteFromClipboard does not retrieve the data in CF_TEXT format. This format is already known by other Windows applications and Delphi components as you will see in the example project.

Using The CF_BIRTHDAY Format

To use this newly defined format, I've created the sample application, shown in Listing 2. The main form is shown in Figure 1. This form simply consists of a TEdit and two TMaskEdit controls into which the user will enter the birthday information. Three additional TEdit controls and one TMemo will receive the information retrieved from the clipboard from the first TEdit and TMaskEdit controls.

Listing 2 shows the code for the main form, which also contains the event handlers for the Copy and Paste buttons.

The CopyBtnClick first creates an instance of TBirthDay called MyBirthDay. It then assigns the initialized TPersonRec variable to TBirthDay's Person property and then calls the method MyBirthDay.CopyToClipBoard. The PasteBtnClick method does the

➤ *Listing 1 The BDAY.PAS unit defining the TBirthDay component*

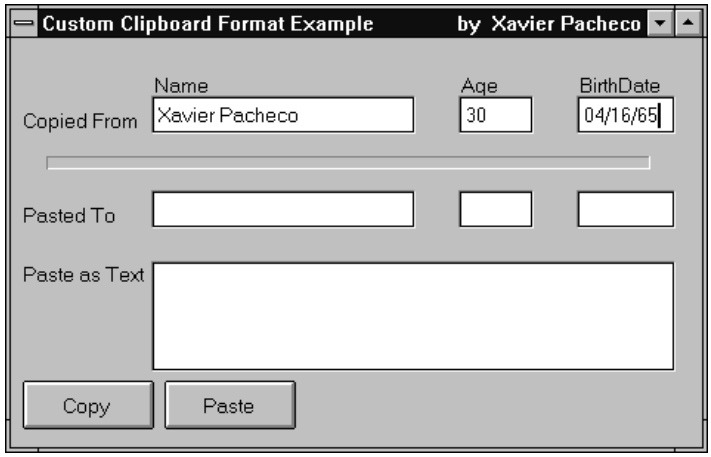
```
unit Bday;
interface
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, ClipBrd;
type
  TPersonRec = record
    Name: string[100];
    Age: integer;
    BirthDate: TDateTime;
  end;
  TBirthDay = class(TComponent)
  private
    { Protected declarations }
    FPersonRec: TPersonRec;
  public
    procedure CopyToClipBoard;
    procedure PasteFromClipBoard;
    property Person: TPersonRec read
       FPersonRec write FPersonRec;
  end;
var
  CF_BIRTHDAY: word;
procedure Register;
implementation
procedure TBirthday.CopyToClipBoard;
const
  CRLF = #13#10;  { Carriage return line feed }
  FormatText = 'Name: %s%sAge: %d%sBirthDate: ';
var
  Data: THandle;
  DataPtr: Pointer;
  Len: Integer;
  TempStr: string;
begin
  { Allocate memory from global heap }
  Data := GlobalAlloc(GMEM_MOVEABLE, SizeOf(FPersonRec));
  try
    DataPtr := GlobalLock(Data);
    try
      { Move the data in Buffer to DataPtr    }
      Move(FPersonRec, DataPtr^, SizeOf(FPersonRec));
      ClipBoard.Open; { This is only required if      }
         { multiple clipboard formats are being saved }
         { at once. Otherwise, if only one format is  }
         { being sent to the clipboard, don't call it }
      try
        ClipBoard.SetAsHandle(CF_BIRTHDAY, Data);
        { Now copy also in the CF_TEXT format also }
        with FPersonRec do
          TempStr := Format(FormatText, [Name, CRLF,
            Age, CRLF])+ DateToStr(BirthDate);
        ClipBoard.AsText := TempStr;
      finally
        Clipboard.Close; { Only call this if you }
            { previously called Clipboard.Open() }
      end
    finally
      { Unlock globally allocated memory }
      GlobalUnlock(Data);
    end;
  except
    GlobalFree(Data);  { Free memory allocated, only }
    raise;             { if an exception occurs as   }
  end;           { this memory is managed by Windows }
end;
procedure TBirthday.PasteFromClipBoard;
var
  Data: THandle;
  DataPtr: Pointer;
  C: Char;
  Size: Integer;
begin
  { Get the data on the clipboard }
  Data := ClipBoard.GetAsHandle(CF_BIRTHDAY);
  try
    { Exit if unsuccessful }
    if Data = 0 then Exit;
    { Lock the Global memory object }
    DataPtr := GlobalLock(Data);
    try
      if SizeOf(FPersonRec) > GlobalSize(Data) then
        Size := GlobalSize(Data);
        { Copy contents of DataPtr to Buffer }
      Move(DataPtr^, FPersonRec, SizeOf(FPersonRec));
    finally
      { Unlock the global memory object }
      GlobalUnlock(Data);
    end;
  except
    GlobalFree(Data);  { Free memory allocated, only }
    raise;          { if an exception occurs as this }
  end;                { memory is managed by Windows }
end;
procedure Register;
begin
  RegisterComponents('Test', [TBirthday]);
end;
initialization
  { Register the special clipboard format CF_BIRTHDAY}
  CF_BIRTHDAY := RegisterClipBoardFormat('CF_BIRTHDAY');
end.
```

opposite. It also creates a TBirthDay instance, then checks for the CF_BIRTHDAY format on the clipboard and calls the method MyBirthDay.PasteFromClipBoard if that format is present. The TEdit controls are updated with the data contained in MyBirthDay.Person whereas the TMemo retrieves that same data in the CF_TEXT format.

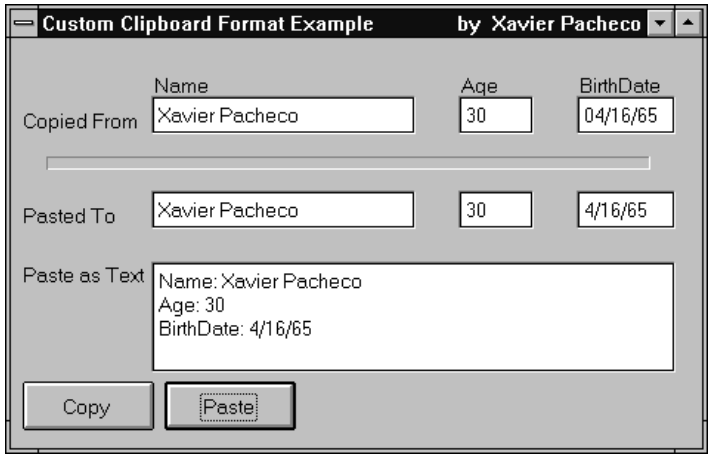
Figure 1 shows the form with data entered before the pasting from the clipboard and Figure 2 shows the form after the data has been pasted from the clipboard.

Xavier Pacheco is a Delphi Developer with TurboPower Software and co-author of *Delphi Developer's Guide* by Sams Publishing. You can reach Xavier on CompuServe at 76711,666

➤ *Figure 1 Main Form before ClipBoard paste*



➤ *Figure 2 Main Form after ClipBoard paste*



➤ *Listing 2 Source code for the main form*

```
unit Persu;
interface
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, BDay, StdCtrls,
  ClipBrd, ExtCtrls, Mask;
type
  TForm1 = class(TForm)
    CopyBtn: TButton;
    NameEdit: TEdit;
    AgePaEdit: TEdit;
    BDPaEdit: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Memo1: TMemo;
    Label3: TLabel;
    PasteBtn: TButton;
    BDMaskEdit: TMaskEdit;
    AgeMaskEdit: TMaskEdit;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Bevel1: TBevel;
    NamePaEdit: TEdit;
    procedure PasteBtnClick(Sender: TObject);
    procedure CopyBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.CopyBtnClick(Sender: TObject);
var
  PersonRec: TPersonRec;
  MyBirthDay: TBirthDay;
begin
  MyBirthDay := TBirthDay.Create(self);
  try
    { Put data from edit controls into PersonRec }
    PersonRec.Name := NameEdit.Text;
    PersonRec.Age := StrToInt(AgeMaskEdit.Text);
    PersonRec.BirthDate := StrToDate(BDMaskEdit.Text);
    { Assign the record to the Person property }
    MyBirthDay.Person := PersonRec;
    { Copy the data to the clipboard }
    MyBirthDay.CopyToClipboard;
  finally
    MyBirthDay.Free;
  end;
end;
procedure TForm1.PasteBtnClick(Sender: TObject);
var
  PersonRec: TPersonRec;
  MyBirthDay: TBirthDay;
begin
  MyBirthDay := TBirthDay.Create(self);
  try
    { Check if format is available }
    if ClipBoard.HasFormat(CF_BIRTHDAY) then begin
      { Get the clipboard data }
      MyBirthDay.PasteFromClipBoard;
      { Copy over the person data }
      PersonRec := MyBirthDay.Person;
      { Update the edit controls }
      NamePaEdit.Text := PersonRec.Name;
      AgePaEdit.Text := IntToStr(PersonRec.Age);
      BDPaEdit.Text := DateToStr(PersonRec.BirthDate);
      { Data in CF_TEXT format also }
      Memo1.PasteFromClipBoard;
    end;
  finally
    MyBirthDay.Free;
  end;
end;
end.
```