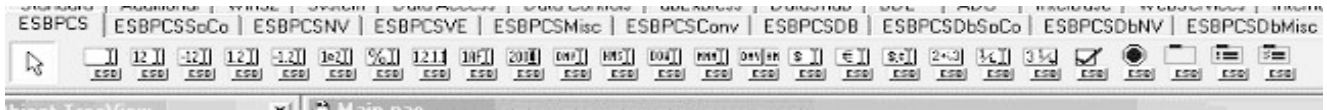


ESBPCS - ESB Professional Computation Suite

product review by John Lee



I was asked a few months ago by BUG if I would share my thoughts about ESBPCS. I received a free copy of the component suite at a BUG meeting, so I thought why not? This will be my way of saying thank you. I actually started using a DateTime library developed by ESB Consultancy that they offered free on the internet. So, when the offer was made at the meeting, I was already familiar with the company.

What is ESBPCS, you say? Well, ESB Professional Computation Suite (ESBPCS), currently at version 2.1.0, is a collection of over 6000 Routines/Methods, over 150 Classes/Components in over 100 units with an emphasis on Data Entry and Manipulation for Developers using Delphi 4, 5 & 6 and Borland C++ Builder 4 & 5.

It provides routines to handle Mathematics, Statistics, Date/Time Manipulation, Time Zones, Financials, Unit Conversions and much more, as well as Components to control Editing and Displaying of Data better. Components include Edits, CalcEdits, Calendar Edits, CheckBoxes, RadioGroups, CheckGroups, Calculators, Calendars, ComboBoxes, SpinEdits, MatrixEdits, VectorEdits, Labels and more. There is a strong emphasis on Data Aware Controls and routines in this collection. It is beyond the scope of this article to cover all of them so I will spend the next couple of pages covering Edit Controls, Calendar Edits, as well as some of the non-visual routine libraries.

ESBPCS tabs

When you install ESBPCS it adds eleven tabs to your component pallet (see above).

The first tab on the pallet is 'ESBPCS'. There are twenty-one variations of the standard edit box. They range from ESBPCSEdit to ESBFractEdit. The standard edit box offers additional features from the standard Delphi edit box as alignment, read only colour, disabled colour, border style (Flat, Underline, Normal). These are settings available from the object inspector in design mode as well as during runtime.

When you begin to use the more specialised edit boxes, you see the advantages they offer. Number edit boxes will only accept keystrokes that provide a numeric input. If you want your users only to enter a positive number, that is covered as well by the ESBPosFloatEdit. You can set the thousand separator, colour of the font for a positive or negative number. The percentage edit box will add a '%' to any number entered. There are edit boxes to handle hexadecimal values, IP addresses and Scientific Float Edit. If you choose to use one of the date/time edit boxes, there again there are checks when you exit the boxes to validate

the value entered. If it is not a valid entry then the value is cleared and the box reverts to its default value or to blank. There is also an ESBPCSDb tab that offers data aware versions of the same edit boxes.

The other tabs installed with ESBPCS are ESBPCSSPCO, ESBPCSNV, ESBPCSVE, ESBPCSMisc and ESBPCSCov. There are also Data Aware versions of these tabs, with the exception of the ESBPCSVE and ESBPCSCov.

I use the Calendar Edit, found on the ESBPCSSPCO tab, frequently and find it very easy. I initially had trouble with the data aware version wanting to interact with the dataset when it was created on the form, but that has been fixed much to my relief in version 2 of the suite. In the past, if I wanted to give my users a graphical method of selecting a date for a database field, I had to use the standard Date Time Picker and then with code copy the selection to a data control.

I have recently started using the Country combobox on the same tab. I needed to supply a country code for each country entered in this database. The country dropdown combobox was great in that it used an XML file to provide the country code as well as other information for the country selected.

A quick look at the xml file will give you an idea of the information stored on each country. As a note though, do make sure you include the xml file in your installation, otherwise you will find for the data aware version of this component, nothing will get displayed when the application opens.

As the name implies, with ESBPCSCov you can very easily provide every possible conversion you can imagine with little effort or lines of code. For me, that is the key to using a component. If using a component reduces the number of lines of code I need to write to accomplish a task then it is worth using.

Financial software

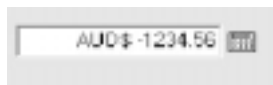
Where ESBPCS really comes into its own is building financial software. Beginning with currency values, this suite offers the developer options as to how to format the currency display, currency conversions, decimal places and more. For example, Delphi's Currency type is a Floating Point Value with a fixed four decimal places. TESBCurrency type is a 32-bit Integer Value with a fixed two decimal places and TESBLongCurrency type is a 64-bit Integer Value with a fixed six decimal places. Let's also take the example of currency symbol format. You can change the appearance of the currency displayed in a TESBLongCurrCalcEdit (found on the ESBPCSSpCo Tab) with the following:

```

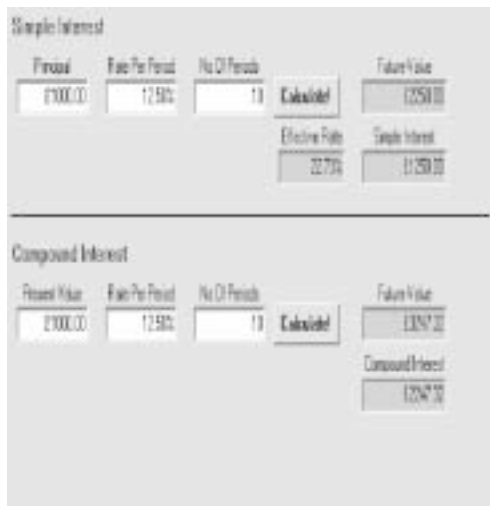
procedure TForm1.CurrFormatRGClick(Sender:
TObject);
begin
    FormatCurr.CurrencyFormat :=
        TCurrencyFormat Type
        (CurrFormatRG.ItemIndex) ;
end

```

This is the result:



Next let's take the process of calculating interest, whether Simple or Compound.



The code to accomplish the simple calculation is as easy as this:

```

procedure TForm1.SICalcBtnClick(Sender:
TObject) ;
begin
    SIInterest.AsESBLongCurrency :=
        SimpleInterest
        (SIPresenValue.AsESBLongCurrency,
        SIRate.AsFloat, SIPeriods.AsLongWord) ;
    SIFutureValue.AsESBLongCurrency :=
        FutureValueSI
        (SIPresentValue.AsESBLongCurrency,
        SIRate.AsFloat, SIPeriods.AsLongWord) ;
    EffectiveRate.AsFloat :=
        EffectiveRateInterest (SIRate.AsFloat,
        SIPeriods.AsLongWord);
end

```

You can see that the values in the boxes labelled 'Future Value', 'Effective Rate' and 'Simple Interest' are supplied by a simple function call. In other words the code is already written: all you need to do is provide the variables needed to complete the calculations. Can't get any simpler than that.

Need to calculate the depreciation of an item? No problem, here is the code;

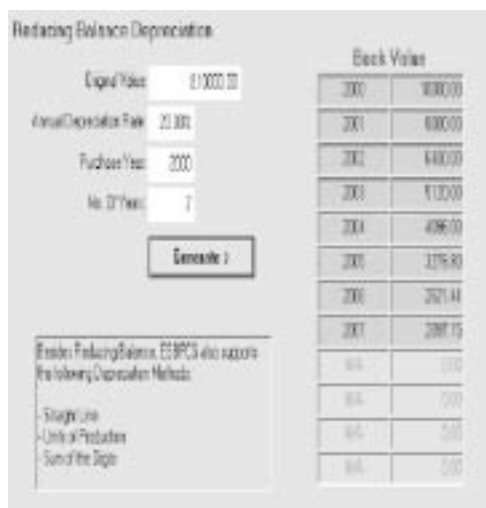
```

procedure TForm1.DepBtnClick(Sender: TObject) ;
var
    StartYr, I, N: LongWord
begin
    StartYr := StartYear.YEar;
    N := NoOfYears.AsLongWord;
    BookValue.NoOfRows := NoOfYears.AsLongWord + 1;
    for I := 0 to N do
        BookValue.Labels [I] := Int2Estr (StartYr + 1);

        BookValue.AsVector := Vector2FloatVector
        (ReducingBalanceDepreciationBookValue
        (OrigValue.AsCurrency, DepRate.AsFloat,
        NoOfYears.AsLongWord) );
end;

```

And here is the output;



Date/time routines

As you would expect, you can calculate a variety of results using their Date/Time routines. These freeware routines are available and supported on the ESBPCS sub-web, and also at <http://www.esbconsult.com.au/downloads.html> along with their other Freeware. They are also included in the full package.

So, do you need to know how many Sundays in March 2002? There is a routine to tell you.

```

Function DOWsInMonth(const DOW: Byte; const
Month, Year: Word): Integer;

```

This function is found in the ESBPCSDateTime unit. You can calculate days apart, weeks apart, convert to ISO value, a day or week within a year. There are so many routines in this unit I can't possible cover them all here.

Scientific apps

If you need to write a scientific application, ESBPCS has you covered here as well. I can't give you any example as I am not familiar with the details of their Complex Numbers Calculations but there is a great demo to give you a start included with the package.

The data aware components work as the non-data aware components do, so there is really no need to go into them in detail here.

An extra bonus for those of you that are not expert programmers, the source code is included with the ESBPCS package so, if you want to learn how they do what they do, you can look at the source code. You can cut and paste to create your own variation of an existing routine.

Closing Comments

I have been using ESBPCS for about 2 years now and I can only scratch the surface of what these components and routines can do to speed up your development process. As I spend more time with this component suite, I find more and more things I can incorporate into my applications, both retail and corporate. A note about the ESBDateEdit box. I tried to change a date in the demo and received an Econvert error while running the demo from the IDE. The error was

an invalid date. However, I did not receive this error during runtime. The nice thing here is that, if you have a problem, the folks at ESB are very good - and I mean very good because I am writing this on a Sunday and received a response to another query today - at answering your questions promptly. The technical support is very good, using web boards and news groups. You must register for the web board and the same username and password will be needed to access the news groups. You can register for the web board here: <http://e-fora.net/~ESB>.

If you are only interested in string or date/time routines and basic arithmetic functions, there is a very good and free library available called JEDI Code Library located here: <http://www.delphi-jedi.org/Jedi:CODELIBJCL>. If you are looking for a more sophisticated set of libraries and components to speed your development of applications that require data validation in edit boxes, financial routines and components, and more, then ESBPCS is worth the cost. As I said earlier, if a component suite saves me time by reducing the number of lines of code I have to write to accomplish a task then it's worth it. The bottom line for all of us is that our time is our inventory and suites like ESBPCS add value to that inventory by doing some of the work for us.

The main site for ESB is <http://www.esbconsult.com.au/esbpcs2>. From here you can get demos and trial versions of the ESBPCS or purchase the product if you choose. There are several version of the software starting from ESBPCS Lite for \$89US. This version does not include the data aware components or routines. The standard single developer licence is \$159US. There are additional versions for multiple developer licences.

John Lee runs Dallas Design Systems, based near Reading. You can contact him on jrlee@ddsystem.com.

Data Entry forms – the easy way

Most database applications need forms to get data from the users. It's quite laborious to create a form from scratch, dropping data-aware components and connecting them to fields in datasets. This is one task that Delphi will do for you, however – this is how:

1. Drop your TTable, TQuery or whatever (preferably on a Data Module) and connect it to your database.
2. Double-click it to display the fields editor.
3. Right-click the fields editor and click “Add All Fields” – Delphi will add a full set of field components to the fields editor.
4. Create a new form.
5. Arrange for the form and the fields editor to both be displayed at the same time.
6. Select all the fields, drag them from the fields editor and drop them on the form. Voila! Delphi will create a data aware component on the form for each of the fields that you drop. It will even add a data source and connect it up.
7. Add an OK button, that calls the dataset's Post method and a Cancel button that calls the Cancel method. When you display the form, put the dataset into Edit or Insert before you display it.

Rob Bracken