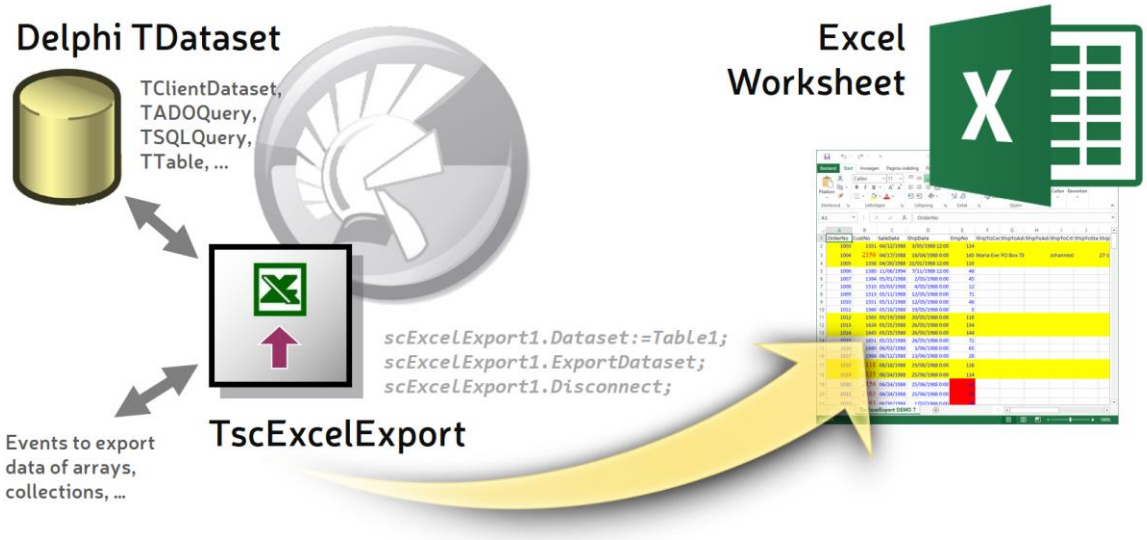
TscExcelExport

VCL component for Delphi

- Support for Delphi 5, 6, 7, 2005, 2006, Turbo Delphi, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle
- Support for Excel 97, 2000, XP, 2003, 2007, 2010, 2013, 2015 and 2016

- Version 4.28
- July 2016

- Stefan Cruysberghs
- Lommel, Flanders, Belgium
- http://www.scip.be



Contents

Description

This TscExcelExport component is an advanced, powerful but easy to use component which enables you to export all records of a dataset from Borland/Codegear/Embarcadero Delphi to Microsoft Excel. Many features are provided to change the layout, use conditional formatting, to add totals, to create groups, to set a filter, ...

This VCL component works in Delphi 5, 6, 7, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle (32 and 64bit). It uses the Microsoft Office type library and it supports Excel 97, 2000, XP, 2003, 2007, 2010, 2013, 2015 and 2016.

Properties

- Name of worksheet and file
- Header and footer texts
- Begin row and column of header, footer, titles (=fieldnames) and data (fieldvalues)
- Width of columns or autofit
- Font of header, footer, titles, summary and data (Alignment, WrapText, Orientation, MergeCells)
- Backgroundcolor and borders of header, titles, summary and data
- Summaries for numeric or given fields (SUM, MIN, MAX, AVG, COUNT)
- Create groups with given fields
- AutoFilter for titles
- A lot of other options to customize the result

Events

- To define the background color and font color, size, name and bold style of each cell
- For exporting data without using a TDataset

Methods

- Export to Excel
- Save worksheet as XLSX (Open XML), XLS, HTML, XML or CSV
- Show print preview

Component editor

- The component editor can be used to change some settings on an easy way.

Live templates

- Live templates for Delphi 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle (ExcelExportUse and ExcelExportCreate)

Type library / COM

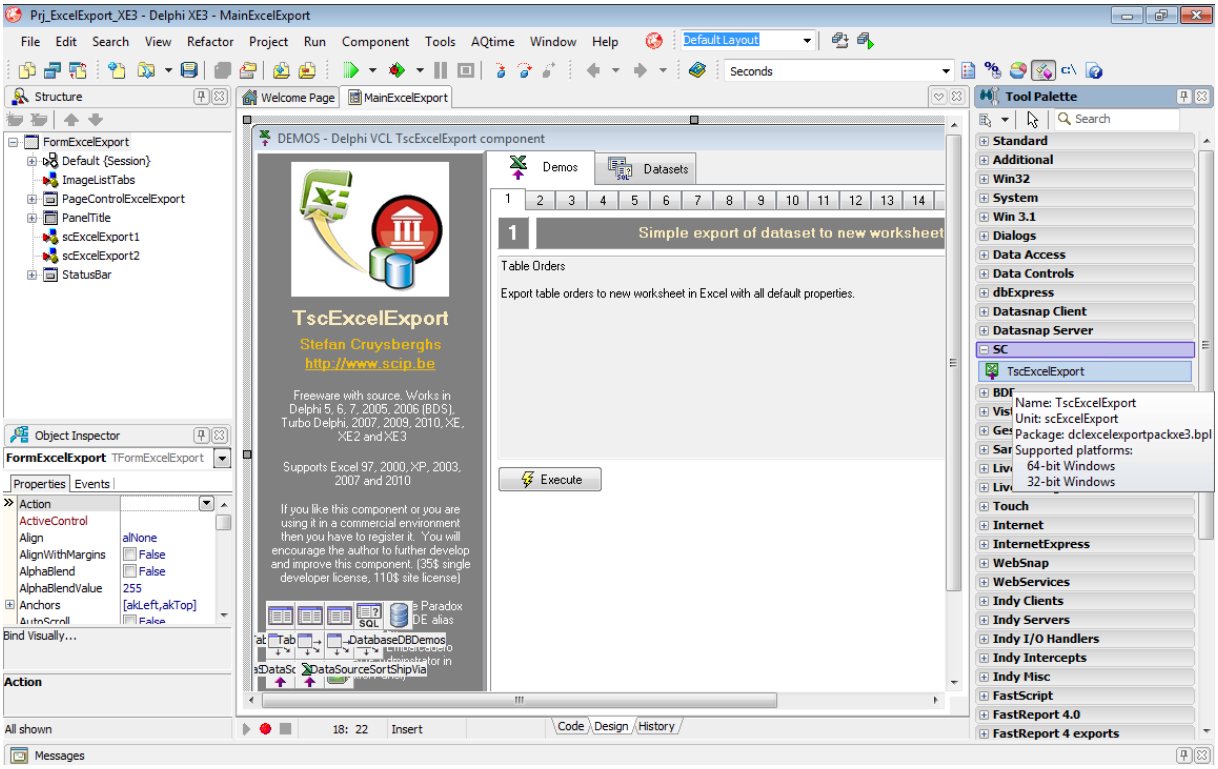
- This component uses the Microsoft Office server components from the Borland COM type library. Each Delphi version supports different Excel type libraries.
- Microsoft Excel should be installed on the PC when using this component.

Delphi / Office

- This VCL component was tested in Delphi 5 (SP1), Delphi 6, Delphi 7, Delphi (BDS) 2005, Delphi (BDS) 2006, Turbo Delphi, Delphi 2007, Delphi 2009, Delphi 2010, Delphi XE, Delphi XE2, Delphi XE3, Delphi XE4, Delphi XE5, Delphi XE6, Delphi XE7, Delphi XE8 and Delphi 10 Seattle
- The TscExcelExport component works with Excel 97, Excel 2000, Excel XP, Excel 2003, Excel 2007, Excel 2010, Excel 2013, Excel 2015, Excel 2016 and it has been tested with Dutch and some English Office versions.
- This component will also give you access to the Excel Application, Workbook and Worksheet objects so you can access all VBA properties and methods.

Demo application

- A full demo application with 18 examples is supplied to view and test all features of the component (and Excel automation).
- See source code for history and more information about properties, methods and events.



Copyrights & registration

- All copyrights to this component are owned by the author Stefan Cruysberghs.
- This component is **freeware for non-commercial** use only and it can be freely distributed.
- The author doesn't give a warranty for error free running of this component.
- If you like this component then you can register it to encourage the author to further develop and improve this component.
- **If you are using the component in a commercial environment/application then you are obligated to register it!**
- Benefits of registering
    - Site license for unlimited developers.
    - Unlimited deployment license.
    - Full source code for Delphi version 5 to XE10.
    - Support will be provided by e-mail.
    - Bugs will be solved as soon as possible.
    - You will receive an email when new versions or upgrades are available.

Prices

- I am using the MyCommerce eSellerate service (http://www.mycommerce.com/esellerate) for payment processing. You can pay safely by credit card online, by fax or by phone.
- Single developer license : **$35** : https://store3.esellerate.net/store/checkout/CustomLayout.aspx?s=STR6765240109
- Site license (unlimited developers) : **$130** : https://store2.esellerate.net/store/checkout/CustomLayout.aspx?s=STR6765240109
- After registering you do not need a registration key. The version which can be downloaded at www.scip.be is fully functional.

Files included

- **scExcelExport.pas**: component
- **scExcelExportReg.pas**: component editor and registration of component
- **scExcelExportConfig.inc**: configuration of type library (97, 2000, XP, 2010)
- **scExcelExport.dcr**: component icon
- **ExcelExportPackx.dpk**: run-time package for Delphi 5, 6, 7, 9 (2005), 10 (2006 & Turbo), 11 (2007), 12 (2009), 14 (2010), XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle
- **dclExcelExportPackx.dpk**: design-time package for Delphi 5, 6, 7, 9 (2005), 10 (2006 & Turbo), 11 (2007), 12 (2009), 14 (2010), XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle.
- **TscExcelExport readme.pdf**: this file
- **DemoExcelExport-2010-X10.exe**: demonstration application
- **DemoExcelExport-5-2009.exe**: demonstration application
- **/Live templates**: the 2 live templates (ExcelExportCreate.xml and ExcelExportUse.xml) for Delphi (BDS) 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 should be copied to
- Delphi 2006
  C:\Program Files\Borland\BDS\4.0\Objrepos\Code_Templates\Delphi
- Delphi 2007
  C:\Program Files\CodeGear\RAD Studio\5.0\ObjRepos\Code_Templates\Delphi
- Delphi 2009
  C:\Program Files\CodeGear\RAD Studio\6.0\ObjRepos\Code_Templates\Delphi
- Delphi 2010
  C:\Program Files\Embarcadero\RAD Studio\7.0\ObjRepos\Code_Templates\Delphi
- Delphi XE
  C:\Program Files\Embarcadero\RAD Studio\8.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE2
  C:\Program Files\Embarcadero\RAD Studio\9.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE3
  C:\Program Files\Embarcadero\RAD Studio\10.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE4
  C:\Program Files\Embarcadero\RAD Studio\11.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE5
  C:\Program Files\Embarcadero\RAD Studio\13.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE6
  C:\Program Files\Embarcadero\RAD Studio\14.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE7
  C:\Program Files\Embarcadero\RAD Studio\15.0\ObjRepos\en\Code_Templates\Delphi
- Delphi XE8
  C:\Program Files\Embarcadero\RAD Studio\16.0\ObjRepos\en\Code_Templates\Delphi

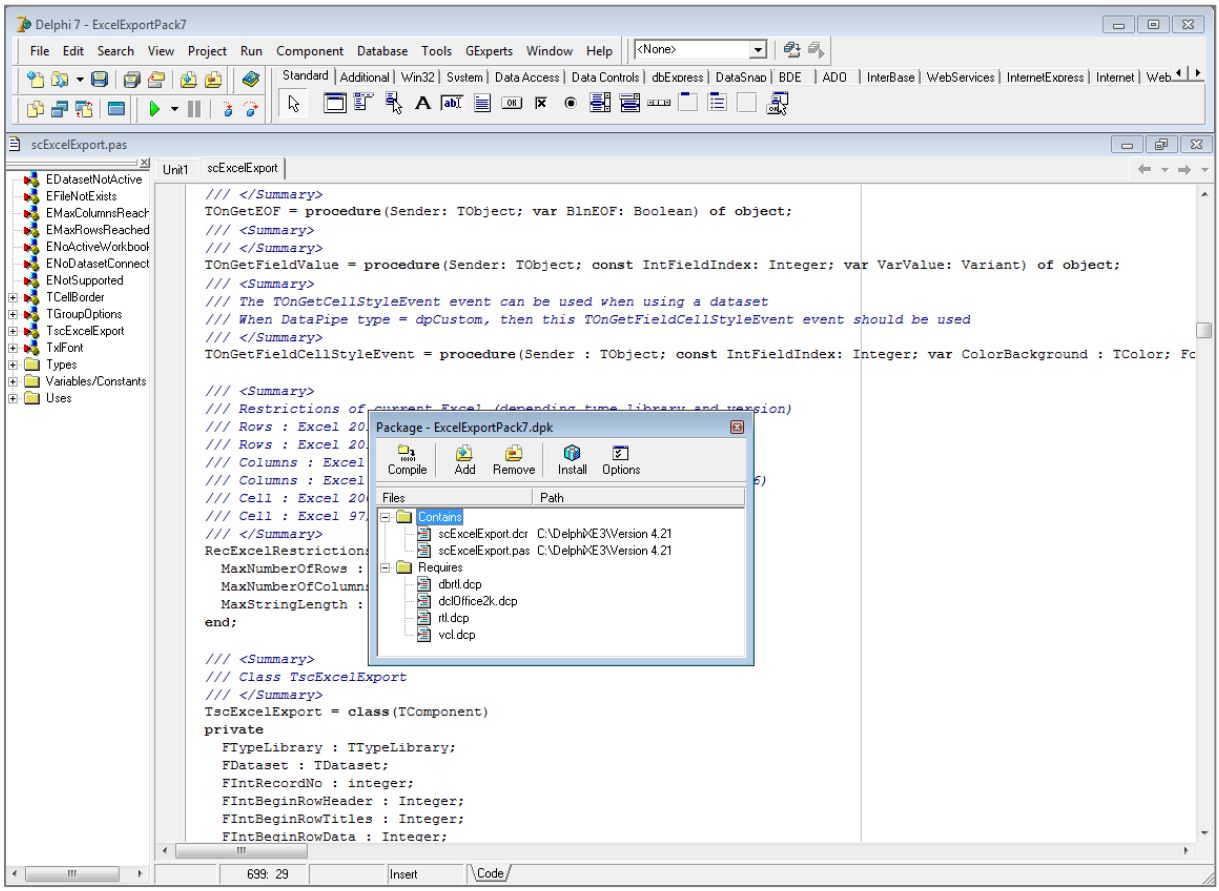
- Delphi XE10
  C:\Program Files\Embarcadero\RAD Studio\17.0\ObjRepos\en\Code_Templates\Delphi
- **/Source demo**: demo project to demonstrate all features of the TscExcelExport component. There is a project for Delphi XE, X2, XE3, XE4, XE5, XE6, XE7, XE8 and 10. The older project can be opened by older Delphi versions but the mainform will give you some errors about properties which do not exist. Just ignore these errors.

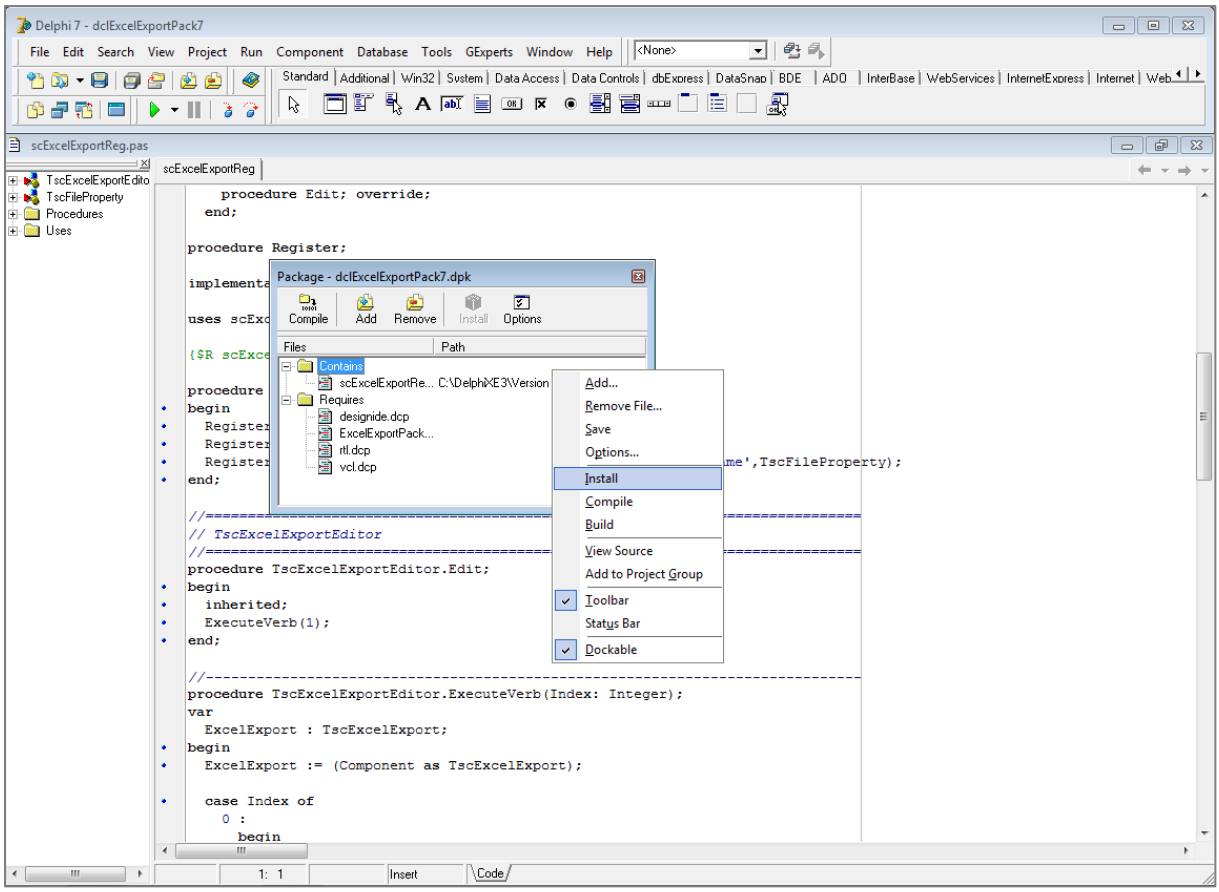
Installation

- Open the run-time package **ExcelExportPackX.dpk** and **build** it
  - Borland Delphi 5 ExcelExportPack5
  - Borland Delphi 6 ExcelExportPack6
  - Borland Delphi 7 ExcelExportPack7
  - Borland Delphi 2005 ExcelExportPack9
  - Borland Delphi 2006 ExcelExportPack10
  - Codegear Delphi 2007 ExcelExportPack11
  - Codegear Delphi 2009 ExcelExportPack12
  - Embarcadero Delphi 2010 ExcelExportPack14
  - Embarcadero Delphi XE ExcelExportPackXE
  - Embarcadero Delphi XE2 ExcelExportPackXE2
  - Embarcadero Delphi XE3 ExcelExportPackXE3
  - Embarcadero Delphi XE4 ExcelExportPackXE4
  - Embarcadero Delphi XE5 ExcelExportPackXE5
  - Embarcadero Delphi XE6 ExcelExportPackXE6
  - Embarcadero Delphi XE7 ExcelExportPackXE7
  - Embarcadero Delphi XE8 ExcelExportPackXE8
  - Embarcadero Delphi 10 Seattle ExcelExportPackD10
  - Embarcadero Delphi 10.1 Berlin ExcelExportPackD101

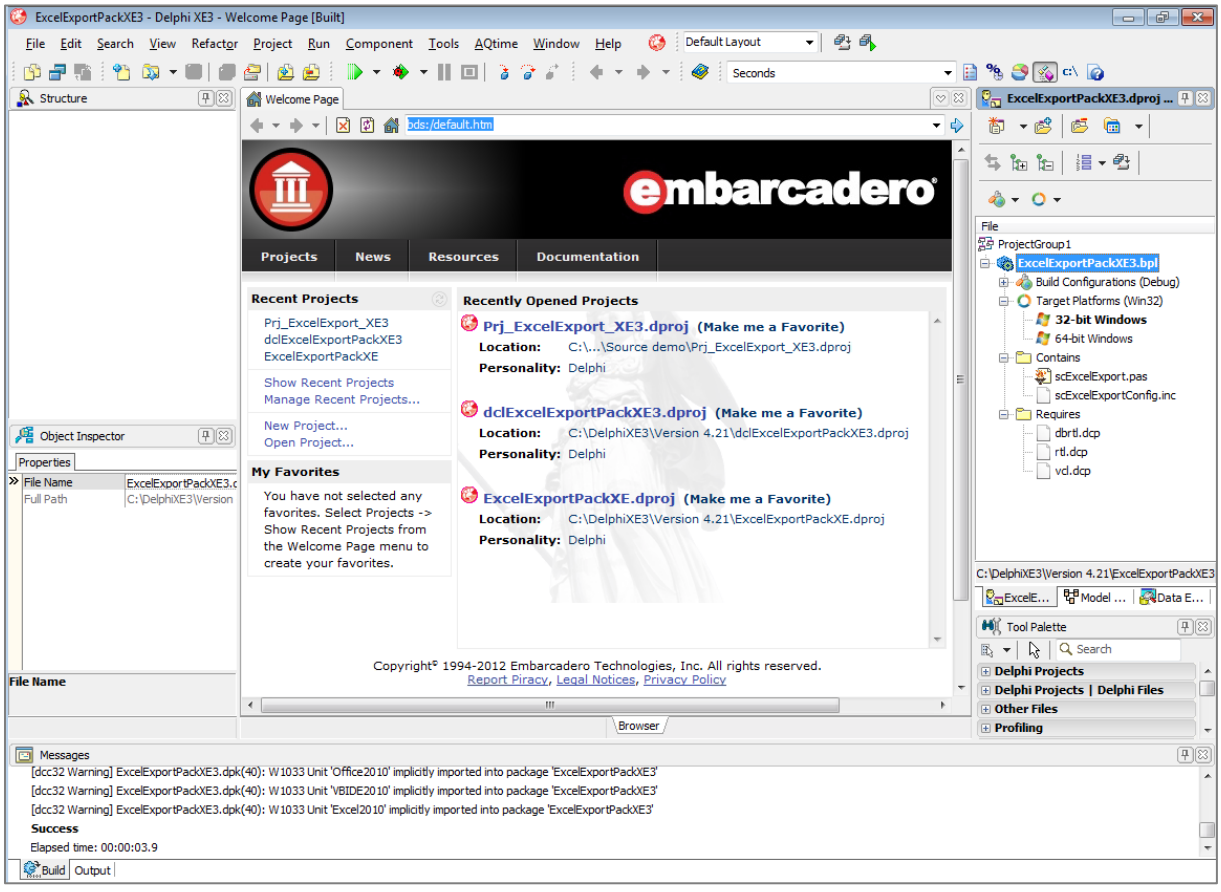
- Open the design-time package **dclExcelExportPackX.dpk** and **compile** and **install** it. The install option is only available in the context popup menu.
- The TscExcelExport component can be found in the tabsheet '**SC**' of the component palette.
- When you like to add the component to an existing package, add the unit scExcelExport to a run-time package. Make sure the DCP file dclOffice is added as required. This file can be found in the Delphi Lib folder. The unit scExcelExportReg.pas contains the registration and property and component editor. This unit should be included in a design-time package.
- The Turbo Delphi (Win32) Explorer edition does not support installing custom components in design time. So the ExcelExportPack10 package cannot be installed. The TscExcelExport component should be created in runtime to use it.
- Make sure to add a path to the folder with the scExcelExport.pas file in the **global Library path** of Delphi (**Tools >> Options >> Delphi Options >> Library**) before compiling the demo project.

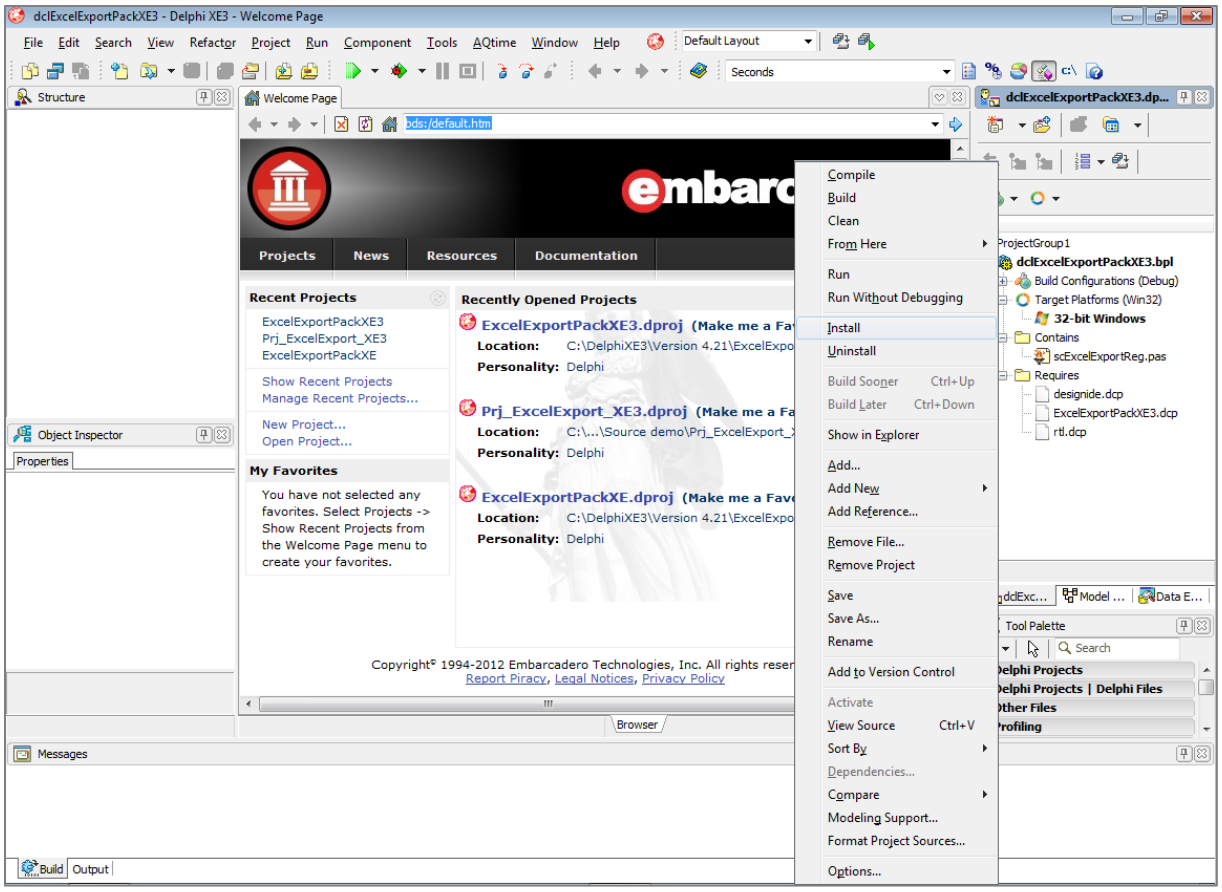
Screenshots Delphi 7

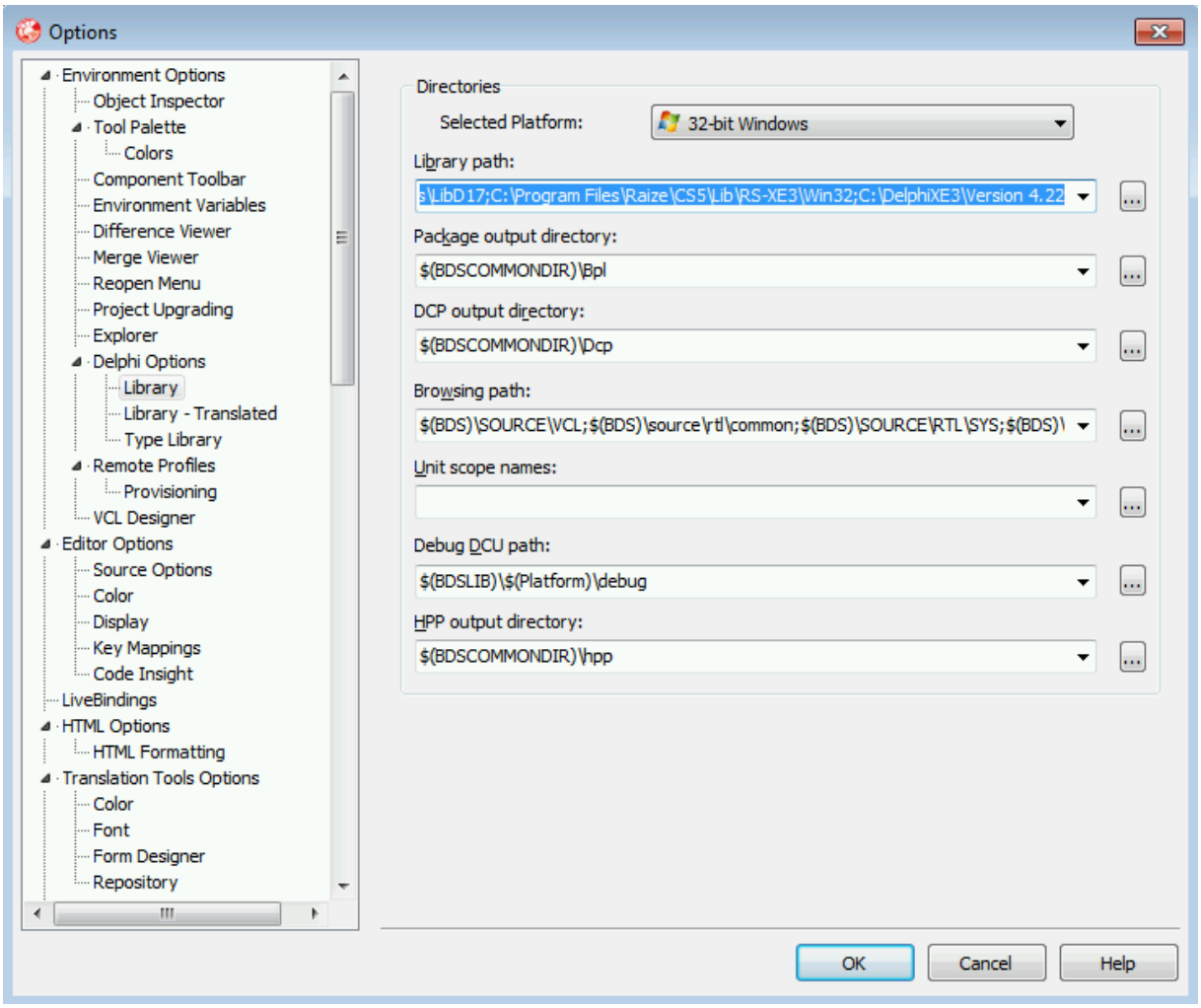




Screenshots Delphi XE3




dclExcelExportPackXE3 - Delphi XE3 - Welcome Page
File Edit Search View Refactor Project Run Component Tools AQtime Window Help
Default Layout
Seconds
Structure
Welcome Page
pds:/default.htm
Projects News Resources Documentation
Recent Projects
ExcelExportPackXE3
Prj_ExcelExport_XE3
ExcelExportPackXE
Show Recent Projects
Manage Recent Projects...
New Project...
Open Project...
My Favorites
You have not selected any favorites. Select Projects -> Show Recent Projects from the Welcome Page menu to create your favorites.
Recently Opened Projects
ExcelExportPackXE3.dproj (Make me a Fa
Location: C:\DelphiXE3\Version 4.21\ExcelExpo
Personality: Delphi
Prj_ExcelExport_XE3.dproj (Make me a Fa
Location: C:\...\Source demo\Prj_ExcelExport_
Personality: Delphi
ExcelExportPackXE.dproj (Make me a Fav
Location: C:\DelphiXE3\Version 4.21\ExcelExpo
Personality: Delphi
Copyright© 1994-2012 Embarcadero Technologies, Inc. All rights reser
Report Piracy, Legal Notices, Privacy Policy
Browser
Object Inspector
Properties
Messages
Build Output
Compile
Build
Clean
From Here
Run
Run Without Debugging
Install
Uninstall
Build Sooner Ctrl+Up
Build Later Ctrl+Down
Show in Explorer
Add...
Add New
Add Reference...
Remove File...
Remove Project
Save
Save As...
Rename
Add to Version Control
Activate
View Source Ctrl+V
Sort By
Dependencies...
Compare
Modeling Support...
Format Project Sources...
Options...
dclExcelExportPackXE3.dp...
ProjectGroup1
dclExcelExportPackXE3.bpl
Build Configurations (Debug)
Target Platforms (Win32)
32-bit Windows
Contains
scExcelExportReg.pas
Requires
designide.dcp
ExcelExportPackXE3.dcp
rtl.dcp
Model ...
Data E...
Tool Palette
Search
Delphi Projects
Delphi Projects | Delphi Files
Other Files
Profiling



Type libraries

What is a type library?

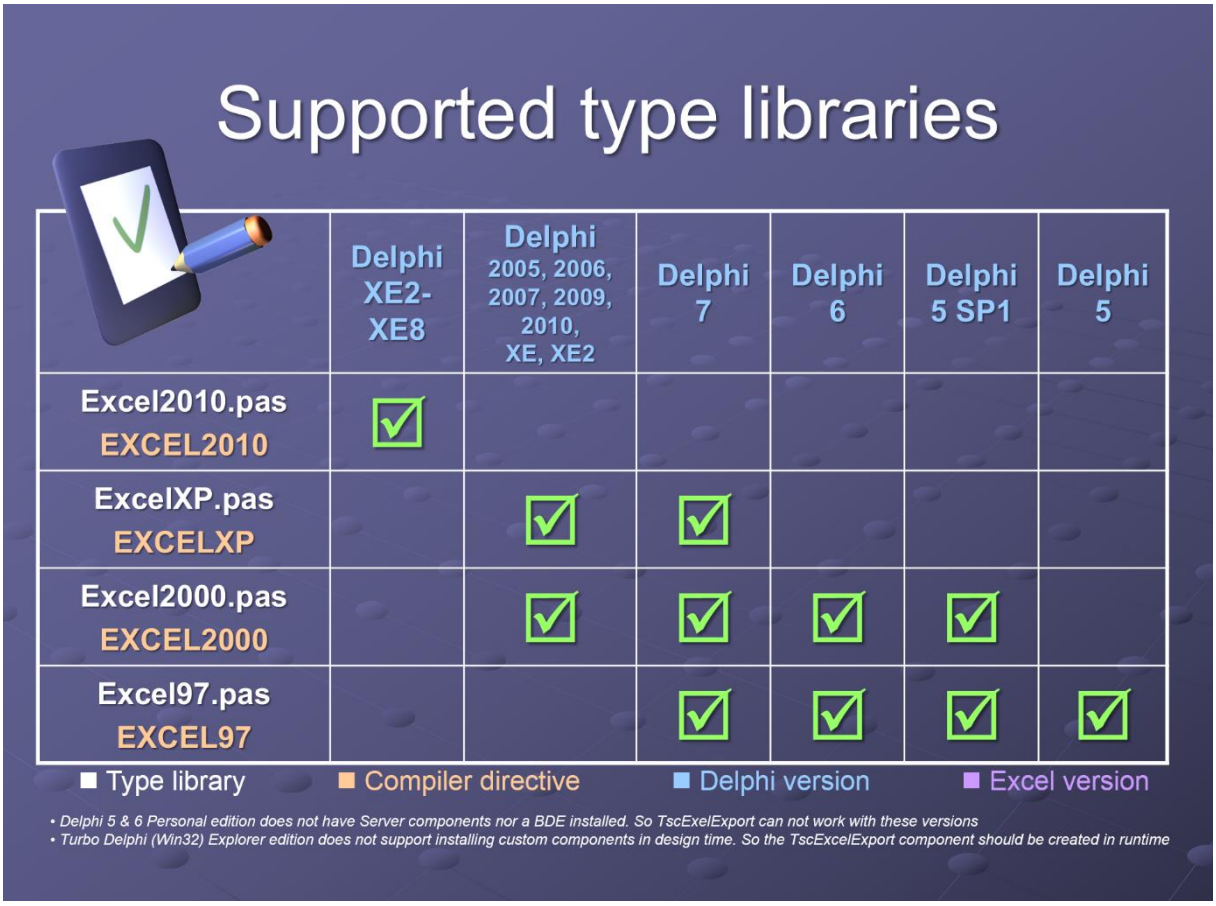
- A type library is a binary file that contains the description about a COM or DCOM object's properties and methods in a form that is accessible to other applications at runtime. These libraries are provided by Microsoft and are included in the Office installation.
- Borland/CodeGear/Embarcadero Delphi provides libraries where the interfaces of these Office objects are converted into an Object Pascal unit file. These Delphi type libraries for the Microsoft Office applications can be found in the **OCX\Servers** folder of your installed Delphi version.
- The TscExcelExport component uses the type library for Excel.
- Each Delphi version supports different Excel type libraries.
- Microsoft Excel should be installed on the PC when using this component.

Configuration in scExcelExportConfig.inc

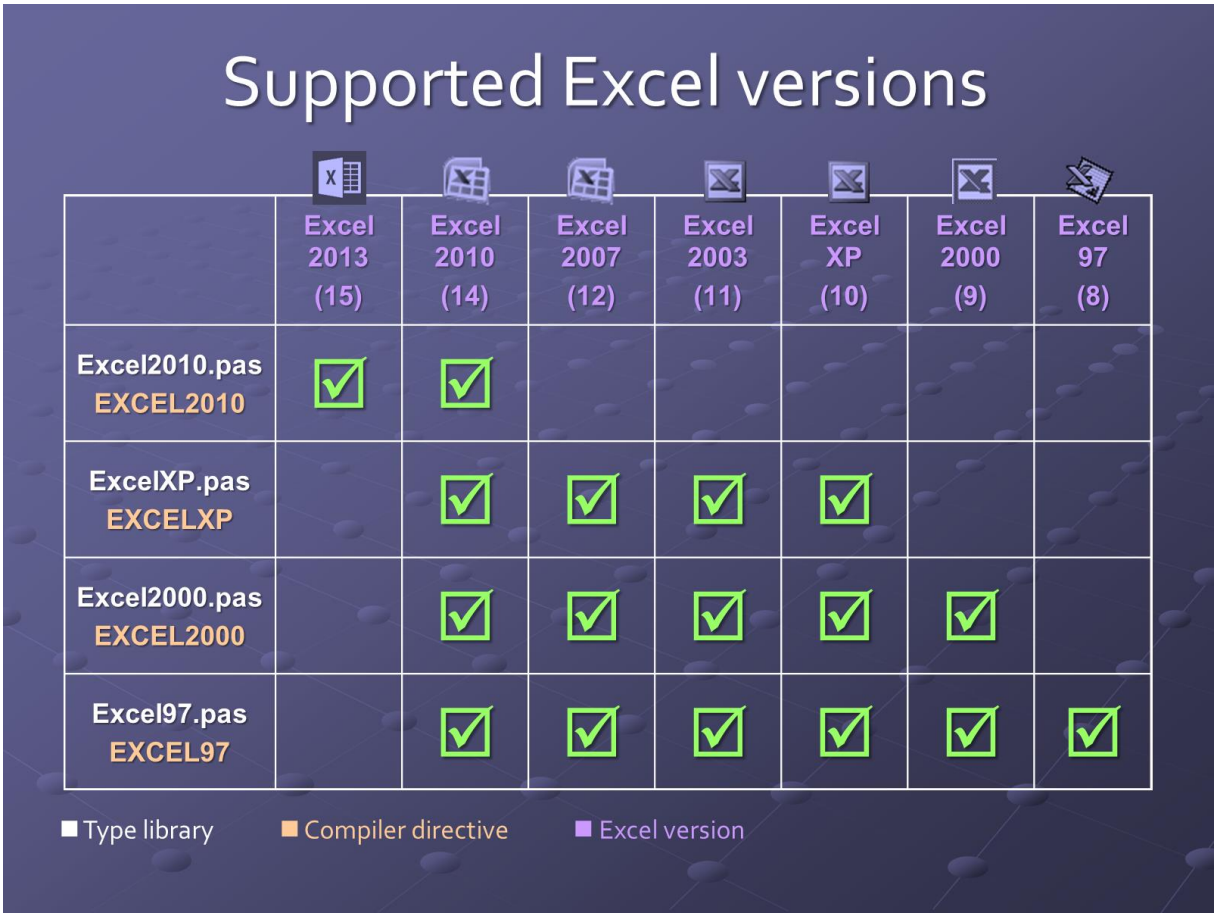
- To override the default type library, you have to uncomment the $UNDEF and $DEFINE lines in the **scExcelExportConfig.inc** file.

```
{-------------------------------------------------------------------------------
* Description : scExcelExportConfig.inc - compiler directives
-------------------------------------------------------------------------------}
{$IFDEF VER280} // Delphi XE7 : Default Excel2010, ExcelXP can also be used
  {$DEFINE EXCEL2010}
  {$DEFINE DELPHIXE7}
  {$DEFINE DELPHIXEORNEWER}
  {$DEFINE DELPHI2010ORNEWER}
  {$DEFINE DELPHI2006ORNEWER}
{$ENDIF}

{$IFDEF VER270} // Delphi XE6 : Default Excel2010, ExcelXP can also be used
  {$DEFINE EXCEL2010}
  {$DEFINE DELPHIXE6}
  {$DEFINE DELPHIXEORNEWER}
  {$DEFINE DELPHI2010ORNEWER}
  {$DEFINE DELPHI2006ORNEWER}
{$ENDIF}
```

Supported type libraries

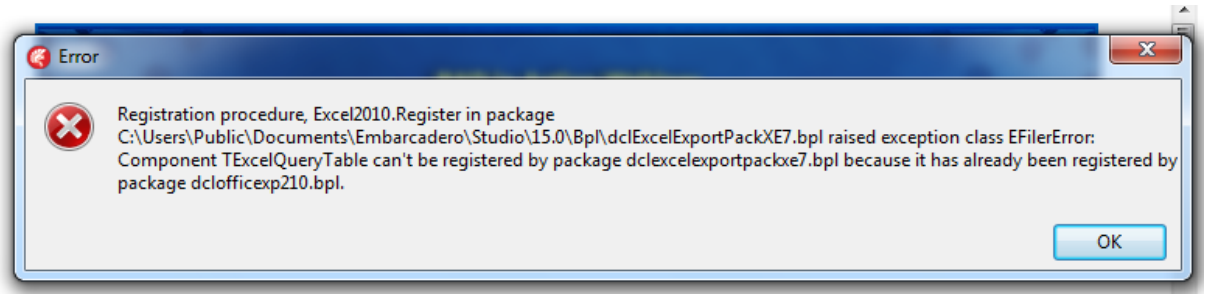


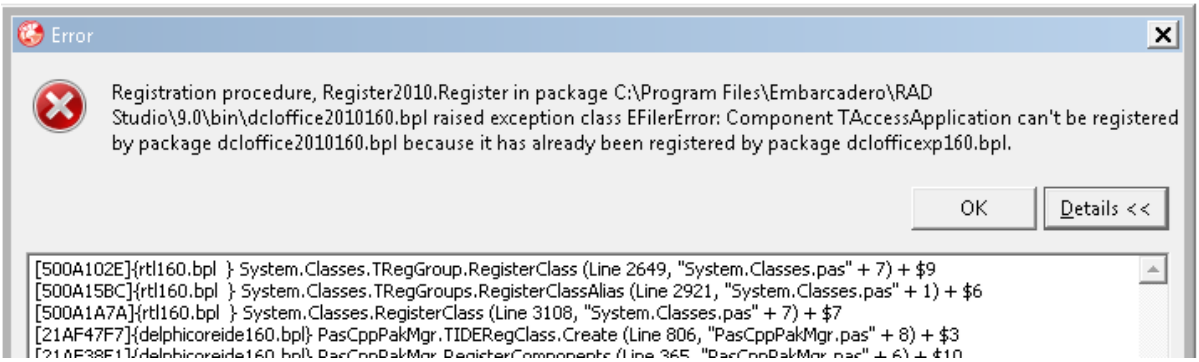
Supported Excel versions



Known issues

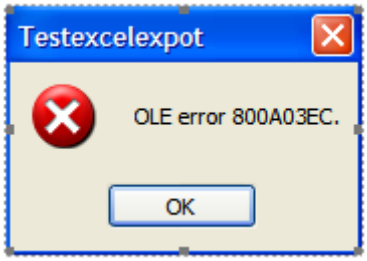
Component can't be registered





This error can occur when installing the design-time package (dclExelExport).It has something to do with another package (of another component) that has already registered one of the Office type libraries provided by Delphi. The solution is to open the TscExcelExport packages but remove all the required libraries of Delphi. Then build and install it. Delphi will add all required packages again and it will skip the ones that are already loaded by other packages.

OLE error / Interface not supported



If your application that exports data to Excel is running perfectly on one computer, but it throws an OLE-error (e.g. 800A03EC) or "Interface not supported" or "Client not registered" exception on an another computer, then there is something wrong with the registration of the COM/OLE libraries (registry reference or missing DLL) of Microsoft Office. To solve it you have to repair or reinstall Microsoft Office.

Demo application in old Delphi version

Error reading ExplicitTop, ExplicitLeft, ExplicitWidth and ExplicitHeight properties when opening the demo application in other Delphi versions than 2007-10.

Just choose "Ignore all". These properties will be removed from the DFM and the application will run fine in Delphi 6, 7, 2006, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10...

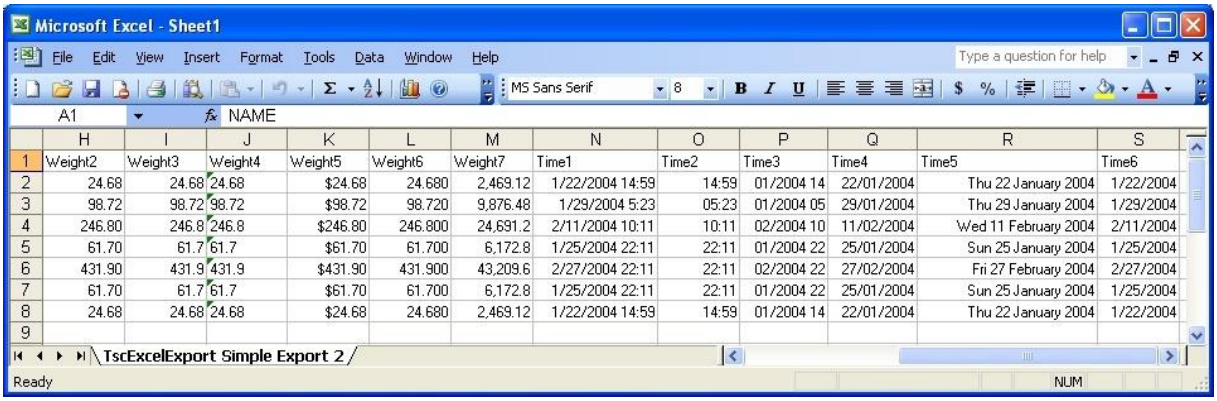
BDE download

The demo application uses the deprecated BDE database technology. The BDE is not included in XE10 anymore so you have to download it from http://cc.embarcadero.com/item/30359

Examples

Example 1: Easiest way to export dataset to Excel

```
scExcelExport1.Dataset:=Table1;
scExcelExport1.ExportDataset;
scExcelExport1.Disconnect;
```



	H	I	J	K	L	M	N	O	P	Q	R	S
1	Weight2	Weight3	Weight4	Weight5	Weight6	Weight7	Time1	Time2	Time3	Time4	Time5	Time6
2	24.68	24.68	24.68	$24.68	24.680	2,469.12	1/22/2004 14:59	14:59	01/2004 14	22/01/2004	Thu 22 January 2004	1/22/2004
3	98.72	98.72	98.72	$98.72	98.720	9,876.48	1/29/2004 5:23	05:23	01/2004 05	29/01/2004	Thu 29 January 2004	1/29/2004
4	246.80	246.8	246.8	$246.80	246.800	24,691.2	2/11/2004 10:11	10:11	02/2004 10	11/02/2004	Wed 11 February 2004	2/11/2004
5	61.70	61.7	61.7	$61.70	61.700	6,172.8	1/25/2004 22:11	22:11	01/2004 22	25/01/2004	Sun 25 January 2004	1/25/2004
6	431.90	431.9	431.9	$431.90	431.900	43,209.6	2/27/2004 22:11	22:11	02/2004 22	27/02/2004	Fri 27 February 2004	2/27/2004
7	61.70	61.7	61.7	$61.70	61.700	6,172.8	1/25/2004 22:11	22:11	01/2004 22	25/01/2004	Sun 25 January 2004	1/25/2004
8	24.68	24.68	24.68	$24.68	24.680	2,469.12	1/22/2004 14:59	14:59	01/2004 14	22/01/2004	Thu 22 January 2004	1/22/2004
9												

Example 2: Using layout properties, adding summary cells and save file

```
scExcelExport1.WorksheetName := 'MyDataset';
scExcelExport1.Dataset:=Table1;
scExcelExport1.StyleColumnWidth:=cwOwnerWidth;
scExcelExport1.ColumnWidth := 20;
scExcelExport1.HeaderText.Text := 'Header';
scExcelExport1.BeginRowHeader := 2;
scExcelExport1.FontTitles := LabelTitle.Font;
scExcelExport1.FontTitles.Orientation := 45;
scExcelExport1.BorderTitles.BackColor := clYellow;
scExcelExport1.BorderTitles.BorderColor := clRed;
scExcelExport1.BorderTitles.LineStyle := blLine;
scExcelExport1.BeginRowTitles := 5;
scExcelExport1.FontData := LabelData.Font;
scExcelExport1.SummarySelection := ssValues;
scExcelExport1.SummaryCalculation := scMAX;
scExcelExport1.ExcelVisible:=False;
try
```

```
  scExcelExport1.ExportDataset;
  if Assigned(scExcelExport1.ExcelWorkSheet) then
    scExcelExport1.ExcelWorkSheet.Range['A1','A10'].Value := 'Delphi';
  scExcelExport1.SaveAs('c:\test.xls',ffXLS); finally
  scExcelExport1.Disconnect;
end;
```

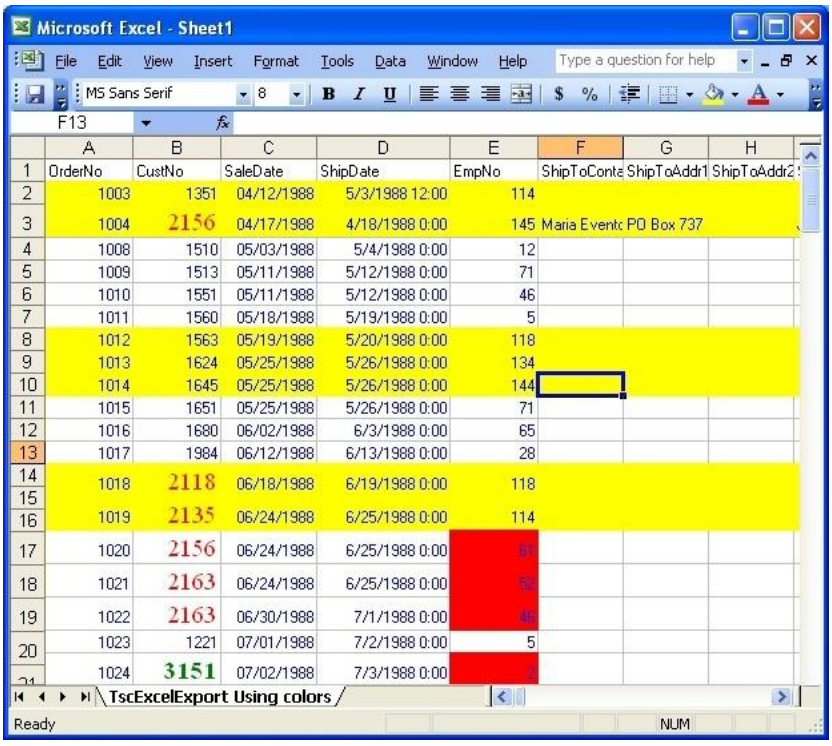
Example 3: Export more datasets

```
scExcelExport1.ExcelVisible:=True;
try
  scExcelExport1.Dataset:=Table1;
  scExcelExport1.WorksheetName:='1';
  scExcelExport1.ConnectTo := ctNewExcel;
  scExcelExport1.ExportDataset;
  scExcelExport1.Disconnect;
  scExcelExport1.Dataset:=Table2;
  scExcelExport1.WorksheetName:='2';
  scExcelExport1.ConnectTo := ctNewWorkbook;
  scExcelExport1.ExportDataset;
  scExcelExport1.Disconnect;
  scExcelExport1.Dataset:=Table3;
  scExcelExport1.WorksheetName:='3';
  scExcelExport1.ConnectTo := ctNewWorksheet;
  scExcelExport1.ExportDataset;
finally
  scExcelExport1.Disconnect;
end;
```

Example 4: Change background color and font style in the OnGetCellStyle event

See full example 7 in demo application

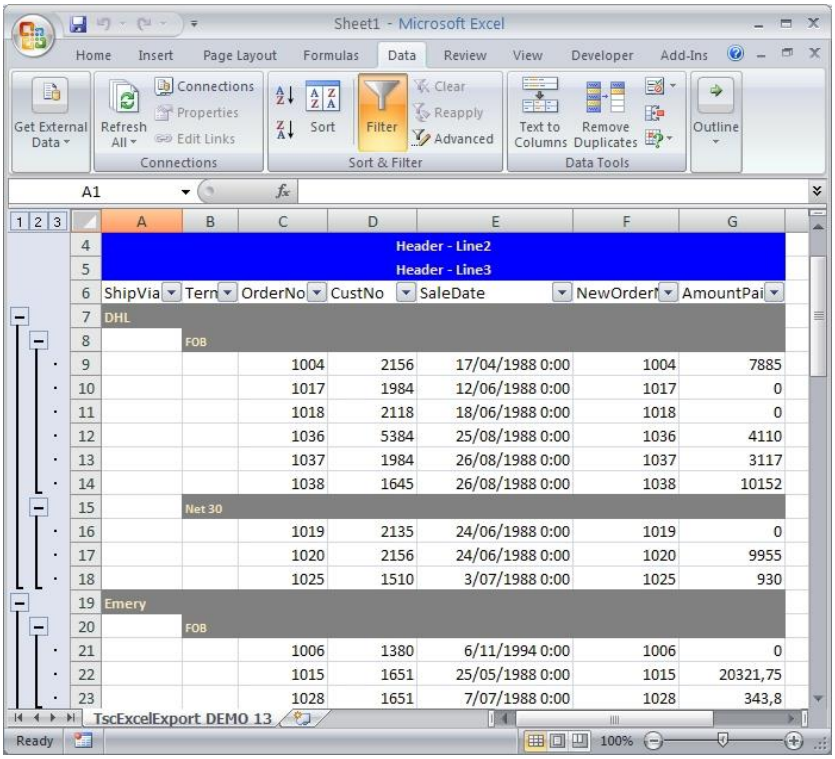
```
procedure scExcelExportGetCellStyleEvent(Sender: TObject; Field: TField;
  var ColorBackground : TColor; FontCell : TxlFont);
begin
  if Field.FieldName = 'CustNo' then
  begin
    if Field.Value > 2000 then
    begin
      FontCell.Color := clRed;
      FontCell.Name := 'Times New Roman';
      FontCell.Size := 14;
    end;
    if Field.Value > 3000 then
    begin
      FontCell.Style := [fsBold];
    end;
  end;
  if Field.FieldName = 'EmpNo' then
  begin
    if Field.Dataset.FieldByName('CustNo').Value > 2000 then
      ColorBackground := clRed;
  end;
  if Field.DataSet.FieldByName('EmpNo').Value > 100 then
    ColorBackground := clYellow;
end;
```



Example 5: Grouping

See full example 13 in demo application

```
try
    scExcelExport1.Dataset:=TableOrders;
    scExcelExport1.GroupFields.Clear;
    scExcelExport1.GroupFields.Add('ShipVia');
    scExcelExport1.GroupFields.Add('Terms');
    scExcelExport1.ExportDataset;
  finally
    scExcelExport1.Disconnect;
  end;
```



Example 6: Export worksheets

See full example 9 in demo application

```
try
  scExcelExport1.LoadDefaultProperties;
  scExcelExport1.ExcelVisible:=False;
  scExcelExport1.WorksheetName := 'TscExcelExport DEMO 9';
  scExcelExport1.Dataset:=TableOrders;
  StatusBar.Panels[1].Text := '';
  scExcelExport1.ExportDataset;

  scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
    +'ExcelExportDefault',ffDefault); //without file extension

  if scExcelExport1.ExcelVersion = 12 then
    scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
      +'ExcelExport2007.xlsx',ffXLSX);

  scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
    +'ExcelExport2003.xls',ffXLS);

  if scExcelExport1.ExcelVersion <> 12 then
    scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
      +'ExcelExport97.xls',ffXL97);

  scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
    +'ExcelExportCSV.csv',ffCSV);

  if scExcelExport1.ExcelVersion >= 10 then
    scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
      +'ExcelExportHTM.htm',ffHTM);

  if scExcelExport1.ExcelVersion >= 11 then
    scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
      +'ExcelExportXML.xml',ffXML);
finally
  scExcelExport1.Disconnect(True);
end;
```

Example 8: Open existing worksheets

See full example 12 in demo application

```
scExcelExport1.LoadDefaultProperties([pgPositions,pgText]);
 scExcelExport1.ExcelVisible:=False;
 scExcelExport1.Dataset:=TableBiolife;
 scExcelExport1.WorksheetName:='Biolife';
 scExcelExport1.Filename:=ExtractFilePath(Application.ExeName)+'ExcelExport.xls';
 scExcelExport1.ExportDataset;
 scExcelExport1.SaveAs(ExtractFilePath(Application.ExeName)
   +'ExcelExport.xls',ffXLS);
 scExcelExport1.Disconnect;
```

Example 9: Access ExcelWorksheet object

See full example 14 in demo application

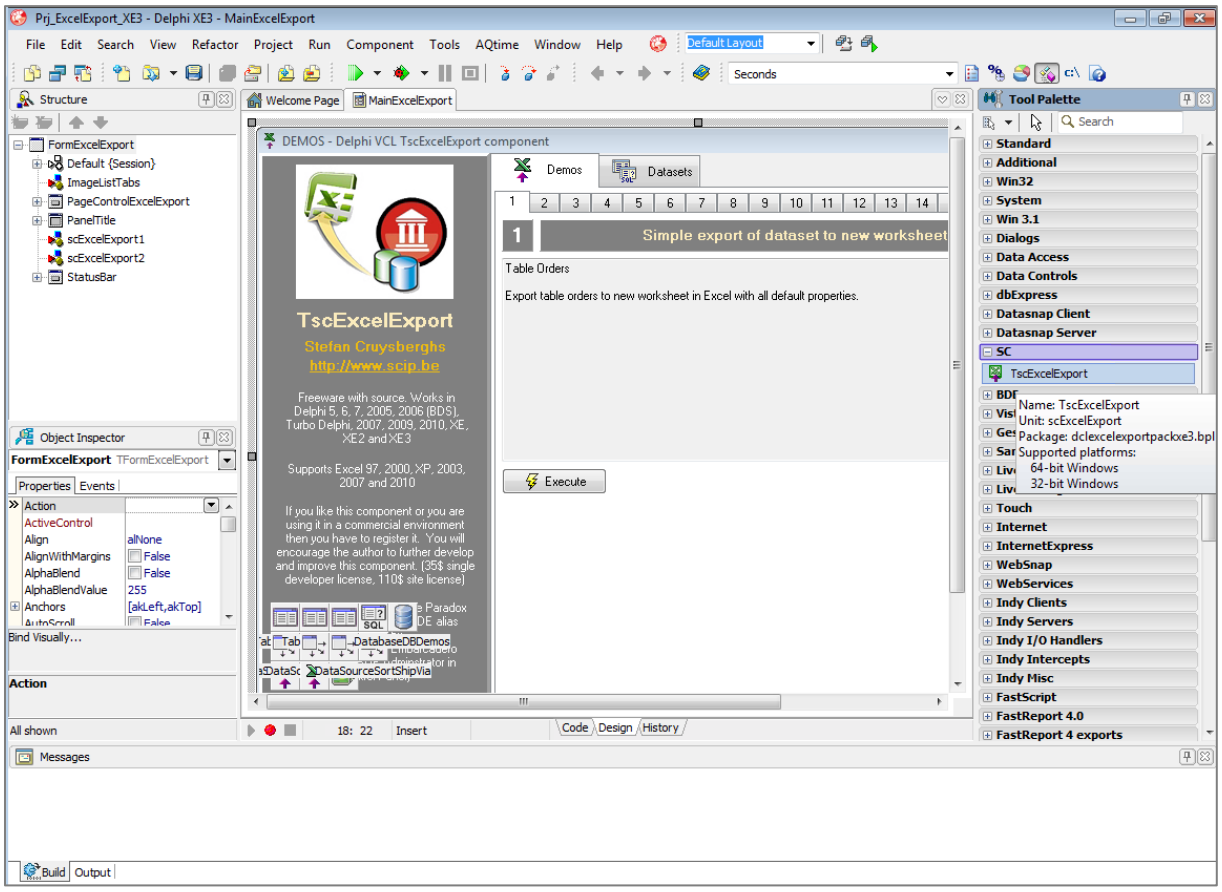
```
try
   scExcelExport1.ExcelVisible:=True;
   scExcelExport1.LoadDefaultProperties;
   scExcelExport1.Dataset:=TableAnimals;
   scExcelExport1.WorksheetName:='TscExcelExport DEMO 14';
   scExcelExport1.ConnectTo := ctNewExcel;
   scExcelExport1.Connect;
   scExcelExport1.ExcelWorkSheet.Range['A2','C8'].Borders.Color := clRed;

   scExcelExport1.ExportDataset;

   scExcelExport1.ExcelWorkSheet.Range['B5','E7'].Cells.Clear;

   scExcelExport1.ExcelWorkSheet.Range[Format('A%d',
     [scExcelExport1.EndRowData+3]),
     Format('A%d',[scExcelExport1.EndRowData+3])].Font.Size := 16;
   scExcelExport1.ExcelWorkSheet.Range[Format('A%d',
     [scExcelExport1.EndRowData+3]),
     Format('A%d',[scExcelExport1.EndRowData+3])].Value2 :=
     'Adding extra information to Excel worksheet';

   scExcelExport1.ExcelWorkSheet.Range['M1','M1'].Value2 := 10;
   scExcelExport1.ExcelWorkSheet.Range['M2','M2'].Value2 := 5;
   scExcelExport1.ExcelWorkSheet.Range['M3','M3'].Value2 := '=M1+M2';
   scExcelExport1.ExcelWorkSheet.Range['M3','M3'].Font.Color := clRed;

   scExcelExport1.ExcelWorkSheet.Range['N1','N20'].Value2 :=
     'Filling extra column with autofit';
   scExcelExport1.ExcelWorkSheet.Range['N1','N20'].Font.Size := 12;
   scExcelExport1.ExcelWorkSheet.Range['N1','N20'].Font.Color := clBlue;
   scExcelExport1.ExcelWorkSheet.Range['N1','N20'].EntireColumn.Autofit;

   scExcelExport1.ExcelWorkSheet.Range['B2','B2'].AddComment(
     'This is comment for a cell');

 finally
   scExcelExport1.Disconnect;
 end;
```

More examples

Check out the demo application with 18 examples which show all the features of the component and how to use the Excel COM type library.

- 1: Simple export
- 2: Calculated fields, display format of numbers and dates
- 3: Memo fields and AutoFilter property
- 4: Memo field and OnGetText event
- 5: Summaries for numeric fields (SummarySelection & SummaryCalculation properties)
- 6: Headers, footers, summaries, fonts & borders (HeaderText, FooterText, BorderHeader, BorderTitles, BorderSummary, BorderData, BorderFooter properties)
- 7: Conditional background color and fonts (OnGetCellStyleEvent)
- 8: Visible fields (VisibleFieldsOnly property)
- 9: Save worksheet to file (XLSX, XLS, CVS, XML and HTML)
- 10: Show print preview
- 11: Add workbooks and worksheets to active Excel instance (ConnectTo & WorksheetName properties)
- 12: Reuse existing files and worksheets (WorksheetName & FileName properties)
- 13: Grouping of data (GroupFields, HeaderText, MergeHeaderCells, FooterText, MergeFooterCells, BorderHeader, BorderTitles, BorderSummary, BorderData, BorderFooter, AutoFilter properties)
- 14: Access ExcelWorksheet object and use all Excel methods and properties (Range, Cells, EntireColumn, Font, AddComment, ...)
- 15: Access ExcelApplication and ExcelWorkbook objects and use all Excel methods and properties (AutoFilter, FreezePanes, WebPagePreview, PivotTableWizard, ...)
- 16: Export data using events instead of dataset (OnGetFieldName, OnGetFieldValue, OnGetFieldCount, OnGetFieldDisplayName, OnGetFieldDataType, OnGotoNextRecord, OnGetFieldCellStyleEvent)
- 17: Close all active Excel instances

- More information about the Microsoft Excel objects can be found at the MSDN website: http://msdn.microsoft.com/en-us/library/7fzyhc74.aspx



Version history

New in version 4.28

- Added support for Delphi 10.1 Berlin
- Bugfix for WideString

New in version 4.27

- Added support for Delphi 10 Seattle
- Added support for Excel 2016

New in version 4.26

- Added support for Delphi XE8

New in version 4.25

- Added support for Delphi XE7

New in version 4.24

- Added support for Delphi XE6
- Bugfix time formatting with AM/PM
- Bugfix currency fields when there are hidden fields

New in version 4.23

- Added support for Delphi XE5

New in version 4.22

- Bugfix for Delphi 2010
- Added support for Delphi XE4
- Unknown Office versions will be handled as new version (number=99)

New in version 4.21

- Bugfixes for Delphi XE3
- Added support for Office 2013

New in version 4.2

- Added support for Delphi XE3

New in version 4.1

- Added support for Delphi XE2. Installing update 1 and 2 of Delphi XE2 is required
- Added support for Excel2010 type library
- Bugfix exporting as XLSX in Office 2010

New in version 4.0

- Added support for Delphi 2010 and XE (2011)
- Added support for Office 2010
- Added support for ftSingle field type
- Bugfix SetFormat when there are invisible fields
- Added COUNT as summary calculation
- Added BackAlternateColor to CellBorder

New in version 3.9

- Added support for Delphi 2010

New in version 3.81

- Support for Interbase TSQLTimeStampField

New in version 3.8

- Improvements to SetAutoFilter method. Now this will also work properly in Excel 97 and 2000
- GetDateTimeFormat has become a public function
- GetCurrencyFormat has become a public function
- Important modifications and bugfixes for exporting dates and times. Now these values are always exported as Delphi datetime (=float) which is the same as the internal Excel datetime value (1 = 1 january 1900). Formatting is done afterwards in the SetFormat function. This should solve all date problems in international Excel versions.
- Added support for Delphi 2009

New in version 3.7

- Added ftWideMemo datatype in IsMemoField function (>= Delphi 2006)
- Added ftFixedWideChar, ftWideMemo, ftOraTimeStamp and ftOraInterval datatype to CanConvertFieldToCell function (>= Delphi 2006)
- Added internal record with Excel restrictions (max rows, max columns, max string length) which will be initialized after checking the Excel type library and version
- Added EMaxColumnsReached and EMaxRowsReached exceptions and resource strings
- Added support for more than 256 columns in Excel 2007
- Bugfix when exporting strings larger than 910 characters in Excel 2003

New in version 3.6

- Added support for Delphi 2007
- Retested support for Excel 2007
- Tested with Windows Vista
- Started with converting comments to XML documentation
- Added public method FindFirstEmptyRow

New in version 3.5

- Added properties for merging of header and footer cells (MergeHeaderCells, MergeFooterCells)
- Added AutoFilter property for titles of dataset
- Added support for Excel 2007 (bèta)
- Added ffXLSX for saving files in Excel 2007 Open XML format
- Added support for OnGetText event for numeric fields
- Added public property for LCID so it can be used after exporting
- Using default font and size of Excel by setting Size = -1 and Font.Name = 'MS Sans Serif' -> Excel97/2003 : Arial size 10, Excel 2007 : Calibri size 11
- Tested with Turbo Delphi Explorer (Win32)
- Renewed demo application !
- Bugfix when setting border and font when using OnGetStyle events

See the source code for information about older versions.