TEkRtf v. 1.85

Properties   Methods   Events   Report template   Example

Unit
EkRtf

Description
TEkRtf report is non visual component that allows you to use all power of MS Word or other rtf-compatible editor to create, preview, edit and print your reports.

How to make it working:

Design time:

- create report template in MS Word
- save it in RTF format
- place TEkRTF component on form or data module
- fill required properties

Run time:

- prepare data in your application - fill property VarList if necessary, prepare Datasets.
- run report using one of Execute methods
- run MS Word (or other editor) if you want to view, edit or print your document

See more details in report template and code examples sections.

TEkRtf Properties

Methods   Events   Report template

InFile
OutFile
Charset
ColorCount
ColorTable
DecimalRSeparator
DecimalRTerminator
DisableControls
ExecuteSuccessful
Lang
LastErrMsg
Options
UDFList
VarList
TrueValue
FalseValue

How to specify DataSets included in report see creating report template and methods Execute, ExecuteOpen.

TEkRtf Methods

Managing variables

VarByName
ClearVars
CreateVar
FreeVar

Executing report

Execute
ExecuteOpen
ExecuteStream
SetTemplateBuffer
FreeTemplate

Other

txt2rtf
ShellOpenFile
Version

Code example

Properties Methods Events

Component TEkRTF is used In this example with properties:
Lang=wdEnglishUS, **Name=InvRTF**, properties **INFILE**, **OUTFILE** filled with corresponding file names. The other properties are left without changing.

There are used tables ORDERS, CUST, ITEMS from DBDEMOS database. In the table ITEMS MasterFields=OrderNo, MasterSource=OrdersSource. Tables are in data module named DM.

Report template:

Send To:

\cust:company\
\cust:addr1\ \cust:addr2\
\cust:city\,\cust:state\ \cust:zip\
\cust:country\

Order No.	**Cust. No.**	**Sales Person**	**Sale Date**	**Ship Date**	**Ship VIA**
\orders:orderNo\	\orders:custNo\	\orders:SalesPerson\	\orders:SaleDate\	\orders:ShipDate\	\orders:shipVIA\

Part No.	**Description**	**Quantity**	**List Price**	**Discount**	**Extended Price**

\scan(items)\

\items:PartNo\	\items:Description\	\items:Qty\	\items:SellPrice\	\items:Discount\%	\items:ExtPrice\

\endscan\

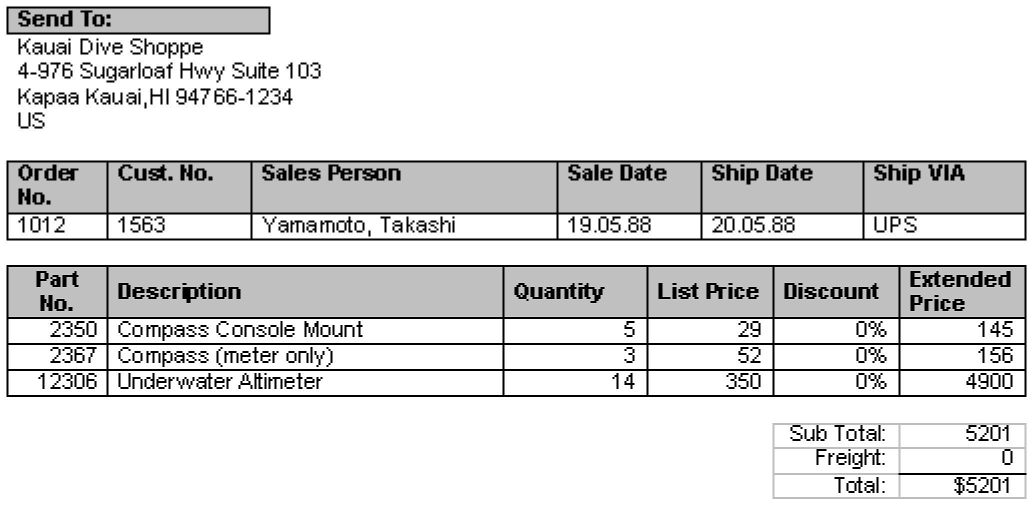
Sub Total:	\Esum\
Freight:	\Freight\
Total:	$\Total\

Before the executing report an user chooses an order number in ORDERS table. Corresponded record in the table becomes the current. Block Scan-Endscan inserts all records from table ITEMS, which are linked with OrderNo from table ORDERS.

Code to generate a report:

```
//Calculate variable Esum
Esum:=CalcExtSum();
//Add variables to varlist
InvRtf.ClearVars;
InvRtf.CreateVar('Esum', Esum);
InvRtf.CreateVar('Freight', Dm.Orders.FieldByName('Freight').AsFloat);
InvRtf.CreateVar('Total', Esum + Dm.Orders.FieldByName('Freight').AsFloat);
//Generate report
InvRtf.ExecuteOpen([Dm.Cust, Dm.Orders, Dm.items], SW_SHOW);
```

Result:



Send To:

Kauai Dive Shoppe
4-976 Sugarloaf Hwy Suite 103
Kapaa Kauai,HI 94766-1234
US

Order No.	Cust. No.	Sales Person	Sale Date	Ship Date	Ship VIA
1012	1563	Yamamoto, Takashi	19.05.88	20.05.88	UPS

Part No.	Description	Quantity	List Price	Discount	Extended Price
2350	Compass Console Mount	5	29	0%	145
2367	Compass (meter only)	3	52	0%	156
12306	Underwater Altimeter	14	350	0%	4900

Sub Total:	5201
Freight:	0
Total:	$5201

See also creating report template, Insert picture example, InsertRtfMemo

TEkRtf font charset

Properties Methods Events

Name	Value
ANSI_CHARSET	0
DEFAULT_CHARSET	1
RUSSIAN_CHARSET	204
OEM_CHARSET	255
SYMBOL_CHARSET	2
MAC_CHARSET	77
SHIFTJIS_CHARSET	128
HANGEUL_CHARSET	129
JOHAB_CHARSET	130
GB2312_CHARSET	134
CHINESEBIG5_CHARSET	136
GREEK_CHARSET	161
TURKISH_CHARSET	162
HEBREW_CHARSET	177
ARABIC_CHARSET	178
BALTIC_CHARSET	186
THAI_CHARSET	222
EASTEUROPE_CHARSET	238

TEkRtf Events

TEkRtf Lang property

Name	Value
WdAfrikaans	1078
WdAlbanian	1052
WdArabic	1025
WdArabicAlgeria	5121
WdArabicBahrain	15361
WdArabicEgypt	3073
WdArabicIraq	2049
WdArabicJordan	11265
WdArabicKuwait	13313
WdArabicLebanon	12289
WdArabicLibya	4097
WdArabicMorocco	6145
WdArabicOman	8193
WdArabicQatar	16385
WdArabicSyria	10241
WdArabicTunisia	7169
WdArabicUAE	14337
WdArabicYemen	9217
WdArmenian	1067
WdAssamese	1101
WdAzeriCyrillic	2092
WdAzeriLatin	1068
WdBasque	1069
WdBelgianDutch	2067
WdBelgianFrench	2060
WdBengali	1093
WdBrazilianPortuguese	1046
WdBulgarian	1026
WdBurmese	1109
WdByelorussian	1059
WdCatalan	1027
WdChineseHongKong	3076
WdChineseMacao	5124
WdChineseSingapore	4100
WdCroatian	1050
WdCzech	1029
WdDanish	1030
WdDutch	1043
WdEnglishAUS	3081
WdEnglishBelize	10249
WdEnglishCanadian	4105
WdEnglishCaribbean	9225
WdEnglishIreland	6153
WdEnglishJamaica	8201
WdEnglishNewZealand	5129
WdEnglishPhilippines	13321
WdEnglishSouthAfrica	7177
WdEnglishTrinidad	11273
WdEnglishUK	2057
WdEnglishUS	1033
WdEnglishZimbabwe	12297

WdEstonian	1061
WdFaeroese	1080
WdFarsi	1065
WdFinnish	1035
WdFrench	1036
WdFrenchCameroon	11276
WdFrenchCanadian	3084
WdFrenchCotedIvoire	12300
WdFrenchLuxembourg	5132
WdFrenchMali	13324
WdFrenchMonaco	6156
WdFrenchReunion	8204
WdFrenchSenegal	10252
WdFrenchWestIndies	7180
WdFrenchZaire	9228
WdFrisianNetherlands	1122
WdGaelicIreland	2108
WdGaelicScotland	1084
WdGalician	1110
WdGeorgian	1079
WdGerman	1031
WdGermanAustria	3079
WdGermanLiechtenstein	5127
WdGermanLuxembourg	4103
WdGreek	1032
WdGujarati	1095
WdHebrew	1037
WdHindi	1081
WdHungarian	1038
WdIcelandic	1039
WdIndonesian	1057
WdItalian	1040
WdJapanese	1041
WdKannada	1099
WdKashmiri	1120
WdKazakh	1087
WdKhmer	1107
WdKirghiz	1088
WdKonkani	1111
WdKorean	1042
WdLanguageNone	0
WdLao	1108
WdLatvian	1062
WdLithuanian	1063
WdMacedonian	1071
WdMalayalam	1100
WdMalayBruneiDarussalam	2110
WdMalaysian	1086
WdMaltese	1082
WdManipuri	1112
WdMarathi	1102
WdMexicanSpanish	2058
WdMongolian	1104
WdNepali	1121
WdNoProofing	1024
WdNorwegianBokmol	1044

WdNorwegianNynorsk	2068
WdOriya	1096
WdPolish	1045
WdPortuguese	2070
WdPunjabi	1094
WdRhaetoRomanic	1047
WdRomanian	1048
WdRomanianMoldova	2072
WdRussian	1049
WdRussianMoldova	2073
WdSamiLappish	1083
WdSanskrit	1103
WdSerbianCyrillic	3098
WdSerbianLatin	2074
WdSesotho	1072
WdSimplifiedChinese	2052
WdSindhi	1113
WdSlovak	1051
WdSlovenian	1060
WdSorbian	1070
WdSpanish	1034
WdSpanishArgentina	11274
WdSpanishBolivia	16394
WdSpanishChile	13322
WdSpanishColombia	9226
WdSpanishCostaRica	5130
WdSpanishDominicanRepublic	7178
WdSpanishEcuador	12298
WdSpanishElSalvador	17418
WdSpanishGuatemala	4106
WdSpanishHonduras	18442
WdSpanishModernSort	3082
WdSpanishNicaragua	19466
WdSpanishPanama	6154
WdSpanishParaguay	15370
WdSpanishPeru	10250
WdSpanishPuertoRico	20490
WdSpanishUruguay	14346
WdSpanishVenezuela	8202
WdSutu	1072
WdSwahili	1089
WdSwedish	1053
WdSwedishFinland	2077
WdSwissFrench	4108
WdSwissGerman	2055
WdSwissItalian	2064
WdTajik	1064
WdTamil	1097
WdTatar	1092
WdTelugu	1098
WdThai	1054
WdTibetan	1105
WdTraditionalChinese	1028
WdTsonga	1073
WdTswana	1074
WdTurkish	1055

WdTurkmen	1090
WdUkrainian	1058
WdUrdu	1056
WdUzbekCyrillic	2115
WdUzbekLatin	1091
WdVenda	1075
WdVietnamese	1066
WdWelsh	1106
WdXhosa	1076
WdZulu	1077

TEkRtf Charset property

Methods   Events   Report template   Properties

```
property Charset: TFontCharset;
```

Sets font charset in output document. This property is usefull for not English reports.
See also: list of charsets

TEkRtf DisableControls property

Methods Events Report template Properties

```
property DisableControls: boolean;
```

If this property set to **true** then method **DisableControls** will be executed in all datasets before processing report template.

Creating report template

Properties Methods Events

You may create a pattern of report by any available editor facilities, using font formatting, justification, colour, tables and other ways of formatting.

All controlling words, variables and data fields must be comprised between symbols "\" (back slash), for instance: **\date** or **\Query1:CustNo**

You may reference to field names using field numbers. For example: **\Query1:(0)**, **\Table1:(5)**

Report generator ignores spaces in field names and keywords. However, if you want to use names with spaces, you may write it between chars "**[**" and "**]**" for example **\Table1:[Field name with spaces]**

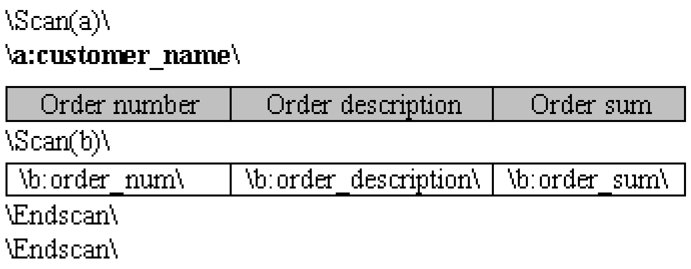
You may create report variable in code with CreateVar method and modify it with VarByName method:
```
EKRTF1.CreateVar('name','Michael');
EKRTF1.VarByName('name').AsString:='John Smith';
```
Variables, will appear in VarList property in format **variablename=value**. For instance: if in the list VarList is kept a line **name=John Smith**, in the pattern of report must be a field **\name**.

Datasets may be identified by name or by means of char **a-z** in that order, in which they were sent in **Execute** method.

In addition to using database fields and variables you may create **user defined functions**, for example **\myfunc(a:field1, a:field2)**. See UDFList property for details.

User defined and format functions may have constant parameters. **Constants** are defined with double quotes, single quotes, or "~" symbols. For example: \"constant1"\, \'constant2' \, \~constant3~\. If you need to place string with "\" symbol in a report template, you also may use a constant, for example: \"c:\My Documents\"\.

You may insert all records of dataset in the document as a table rows or in any free form. For this use keywords **\Scan(datasetname)** and **\endscan**. Inside cycle scan-endscan may be located block of text with data fields and variables, for example:


\Scan(a)\
\a:customer_name

Order number	Order description	Order sum

\Scan(b)\

\b:order_num\	\b:order_description\	\b:order_sum\

\Endscan\
\Endscan\

Lines with words "scan", "endscan" are excluded from the result document. However, if in step of designing a report you want to see as will look a result, you may set an attribute "hidden font" for words scan, endscan.

Full format of scan block is:

\Scan(DataSet) [, while(UDF(...))] [,page] [,noeof] [,function1,...,functionN]\
....

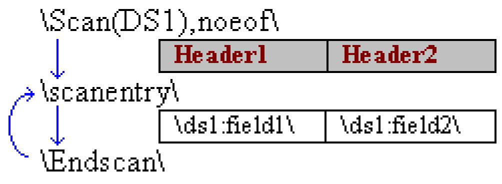
\Scanentry [,function1,...,functionN]\
....
\Scanfooter [,function1,...,functionN]\
....
\Endscan [,function1,...,functionN]\

If keyword **"while"** defined in scan expression, scan block will be terminated when user function UDF(...) returns **false** result. "While" is often used with records grouped by some data field.

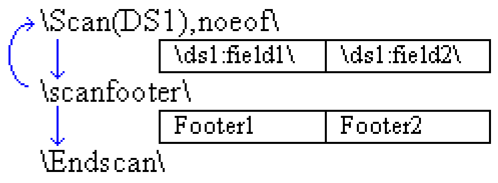
Option "**page**" forces to begin every record of scanned dataset (besides first) from new page.
If you use option **"noeof"** report generator will skip entire scan block if scanned **DataSet** will have no any records. This option is useful when making master-details reports.

Words "Scanentry" and "Scanfooter" are optional. You may add them when using option "noeof" in "scan" keyword, or if you want to make some special functionality calling optional scan block functions.

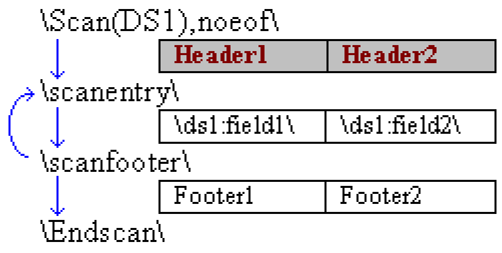
Use option "noeof" with keyword **\Scanentry** to manage scan block with some header section. Every new record of dataset will return control to the position of \Scanentry\ keyword. However, if dataset has no any records, entire block from "scan" to the "endscan" will be missed. For example:



You may use keyword **\Scanfooter** to manage scan block with some footer section. Every time when report generator gets "Scanfooter", it returns control to the position of \Scanentry\ or \Scan\ keyword. If dataset has no any records, entire block from "scan" to the "endscan" will be missed. For example:



\Scanentry\ and \Scanfooter\ may be used simultaneously:



NOTE: You must type keywords \scan(datasetname)\, \scanentry\ and \endscan\ all with the same format attributes, for example with font Arial, 10, regular (or other that you like). It guarantees that format attributes inside block scan-endscan will be correct in output document.

If you use "page" option and table immediately after "scan" keyword in report template, keep in mind that you should have at least one paragraph (empty line) before the table in RTF document, otherwise, RTF editor such as MS Word ignores "new page" control.

See also code example, functions in scan block commands, format functions

Limitation of this version:

Cycles scan - endscan must be outside of tables.

TEkRtf Lang property

Methods   Events   Report template   Properties

```
property Lang:Word;
```

Sets language identifier in output document. Default value=wdLanguageNone.
See also: list of languages

TEkRtf VarList property

```
property VarList:TStrings;
```

This property contains list of variables included in report. Format of strings is **variablename=value**. For example - if you have variable **\date** in report template, you must add string in VarList anything like **date=01/10/2000**. If you have variable **\name** in your report, string in VarList will be like **name=John Smith**.

See also CreateVar method

TEkRtf TrueValue property

```
property TrueValue:string;
```

Value of this property will come up in fields of type boolean if they will be **True**.
For example, if property **TrueValue='Yes'** and data field **Query1.field1=True** then field **\Query1:field1** in report will be **Yes**.

See also FalseValue, Format functions

TEkRtf FalseValue property

property FalseValue:string;
Value of this property will come up in fields of type boolean if they will be **False**.
For example, if property **FalseValue='No'** and data field **Query1.field1=False** then field **\Query1:field1** in report will be **No**.

See also TrueValue, Format functions

TEkRtf Format functions

You may use different format functions and properties depending on data type that you want to place in your report:

Numbers - fexp, ffix, fnum, ffixr, fnumr, fcur functions.

Dates - fdtm function

Graphics - fimg function, OnImageFormat event.

Hyperlink - flnk function.

Boolean - TrueValue, FalseValue properties.

See also creating report template.

TEkRtf OnFinished event

```
property OnFinished : TNotifyEvent;
```

Event OnFinished appears after performing the methods Execute, ExecuteOpen.
In the procedure of processing a given event you may insert a code for some additional actions.

```
procedure TForm1.EkRtf1Finished(Sender: TObject);
begin
showmessage('report finished');
//Here you can do something
end;
```

OnFinished
OnImageFormat
OnScanBefore
OnScanRecord
OnScanEof

TEkRtf OnImageFormat event

```
type TEkOnImageFormat = procedure(FormatIndex:integer; var
ImageFormat:TEkImageFormat) of object;
```

```
property OnImageFormat:TEkOnImageFormat;
```

Event OnImageFormat occurs when report generator is ready to put graphic image in result document. It happens when data field of type ftGraphic presents in report template or when you use function **fimg** to out data field or variable as graphic image.

FormatIndex is parameter from function **fimg** (0 by default). **ImageFormat** is the default format for graphic image. You may change it in given event before the image will be putted in report.

```
procedure TForm1.EkRTF1ImageFormat(FormatIndex: Integer;
  var ImageFormat: TEkImageFormat);
begin
 with ImageFormat do
  begin
    Proportional:=true;
    case FormatIndex of
      1: ScaleX:=50;
      2: FitScaleToY(120);
    end;
  end;
end;
```

TEkImageFormat

TEkImageFormat Properties   TEkImageFormat Methods

Unit
EkRtf

Description
Use TEkImageFormat to set scale and border properties for graphic images in report.
TEkImageFormat Properties   TEkImageFormat Methods

See also OnImageFormat event

TEkRtf Format graphics

Methods   Events   Report template   Properties   Format functions

You may insert graphic images in report from blob, graphic fields and from external graphic files. For this use function **fimg**. For example: **\fimg(a:field2)**, **\fimg(a:field2,1)**.

Function format for blob and graphic fields is: **fimg(FieldName,[FormatIndex])**\, where **FieldName** is name of data field. **FormatIndex** is number that will be passed into OnImageFormat event. This parameter is 0 by default and is not required. You may use this number in event OnImageFormat when formatting different images in report.

Function format for variables is the same: **fimg(VarName,[FmtNumber])**\. Variable VarName must contain string with filename to be inserted in the report. For example: if variable **MyPicture='c:\pictures\chart1.bmp'**, in the report template may be string **\fimg(MyPicture)**. String constant is also allowed as argument for fimg format.

You may insert graphics through the user defined function. For details see TEkUDF TPicture example

See also OnImageFormat event

TEkRtf InFile property

Methods   Events   Report template   Properties

`property Infile:TFileName;`

The name of input RTF file with report template. Required for Execute, ExecuteOpen methods.

See also OutFile property

TEkImageFormat properties

TEkImageFormat   TEkImageFormat Methods

property `Border:TEkImageBorder;`
Represents border type, width and color for given image. By default images putted in the report have no borders.

property `SizeXmm:Double;`
Width of image in millimeters. Set width of image in OnImageFormat event. This property is recommended for using when you need to set exact size of graphic in report.

property `SizeYmm:Double;`
Height of image in millimeters. Set height of image in OnImageFormat event. This property is recommended for using when you need to set exact size of graphic in report.

property `SizeX:Word;`
Width of image in pixels. Read only.

property `SizeY:Word;`
Height of image in pixels. Read only.

property `ScaleX:Word;`
Image scale X in percent. 100 by default.

property `ScaleY:Word;`
Image scale Y in percent. 100 by default.

property `Proportional:Boolean;`
If this property is set to true, then changing ScaleX will change proportional ScaleY, changing ScaleY will change proportional ScaleX. If you want to set ScaleX different to ScaleY - set proportional=false first.

See also OnImageFormat event

TEkImageBorder

Unit
EkRtf

Description

```
type TEkImageBorder=record
               BrType:TEkImageBorderType;
               Width:Single;
               ColorIndex:Word;
               end;
```

TEkImageBorder represents border properties for graphic images in report.
BrType is border type, **width** - border width in points (0.25-3.5), **ColorIndex** - Index from ColorTable (0..ColorCount).

See also OnImageFormat event

TEkRtf ColorCount property

Methods Events Report template Properties

`property ColorCount:Word;`

Number of colors in color table that report generator adds to output rtf document. Read only.

See also: ColorTable property, TEkImageBorder

TEkImageBorderType

Unit
EkRtf

Description

```
type TEkImageBorderType=0..6;
```

Describes border type for images in report. You may use constants defined in unit EkRTF for variables of this type.

constant	value	description
brNone	0	No borders.
brSingle	1	Single border.
brDouble	2	Double border.
brThick	3	Thick border
brShadow	4	Shadow border
brDot	5	Dotted border
brHair	6	Hairline border

See also OnImageFormat event, TEkImageBorder, TEkImageFormat

TEkRtf ColorTable property

Methods Events Report template Properties

type TEkColorArray=array of TEkColor;

property ColorTable:TEkColorArray;
Each RTF document may contain table of colors used in document body. In addition RTF generator adds 16 color table entries to output document. These color entries may be redefined before generating report. For example:

```
EkRtf1.ColorTable[0].color:=clAqua;
```

Colors are used in methods like TEkImageFormat.SetBorderType

Color	Red	Green	Blue
0	$00	$00	$00
1	$00	$00	$FF
2	$00	$FF	$FF
3	$00	$FF	$00
4	$FF	$00	$FF
5	$FF	$00	$00
6	$FF	$FF	$00
7	$FF	$FF	$FF
8	$00	$00	$80
9	$00	$80	$80
10	$00	$80	$00
11	$80	$00	$80
12	$80	$00	$00
13	$80	$80	$00
14	$80	$80	$80
15	$C0	$C0	$C0

See also: ColorCount property, TEkImageBorder

TEkImageFormat methods

TEkImageFormat   TEkImageFormat Properties

constructor `create(x,y:Word);`
Use Create to programmatically instantiate a TEkImageFormat object. Create sets: SizeX=x, SizeY=y, ScaleX=100, ScaleY=100, Proportional=true.

procedure `FitScaleToX(x:word);`
Sets such scale that width of image was equal x pixels.If proportional=true ScaleY also will be changed.

procedure `FitScaleToY(y:word);`
Sets such scale that height of image was equal y pixels.If proportional=true ScaleX also will be changed.

procedure `SetSizeXY(x,y:word);`
Sets SizeX=x, SizeY=y, ScaleX=100, ScaleY=100. This method do not changes actual size of image. It only needs for size reinitialization of TEkImageFormat instance.

procedure `SetBorderType(BrType:TEkImageBorderType; BrWidth:Single; ColorIndex:Word);`
Sets image border with specified parameters.

See also OnImageFormat event

TEkColor

TEkColor Properties   TEkColor Methods

Unit
EkRtf

Description
TEkColor is used to specify the color values. It has properties to operate with color value as standard Delphi TColor type and as its red, green, blue parts separately.

See also ColorTable

TEkColor properties

TEkColor   TEkColor Methods

property `r:byte;`
Red intensity of color.

property `g:byte;`
Green intensity of color.

property `b:byte;`
Blue intensity of color.

property `color:TColor;`
Represents color as standard Delphi TColor type.

TEkColor methods

TEkColor   TEkColor Properties

`constructor create;`

Use Create to programmatically instantiate a TEkColor object. Create method sets R=0, B=0, G=0 that equals clBlack color value.

TEkRtf OutFile property

Methods Events Report template Properties

property `OutFile:TFileName;`

The name of output RTF file that contains result document generated with Execute or ExecuteOpen methods.

See also InFile property

TEkRtf Options property

Specifies various display and behavioral properties of the report.

```
type
TEkRTFOption=(eoGraphicsWmfCompatible, eoGraphicsBinary, eoClearMissedFields,
eoDotAsColon, eoNumericFormatClearZero);
TEkRTFOptions=set of TEkRTFOption;

property Options:TEkRTFOptions;
```

Description
Set Options to include the desired properties for the report.

eoGraphicsWMFCompatible - Graphics inserted in report will be in WMF compatible format. Set this option if you use WordPad or other free and shareware RTF editors. You may clear this option when using MS Word editor.

eoGraphicsBinary - Graphics inserted in report will be in binary format. For using with most free and shareware RTF editors you must clear this option. Your report will be compatible with most editors, but file with graphic picture will be very large size, because graphics inserted in report will be in hexadecimal text format. You may set this option when using MS Word editor.

To get smallest RTF file set eoGraphicsWMFCompatible to false, eoGraphicsBinary to true. To get largest, but compatible with most editors RTF file set eoGraphicsWMFCompatible to true, eoGraphicsBinary to false.

eoClearMissedFields - Field, inserted in report template, will be deleted if RTF generator cannot find it's name in VarList and if name of the field is not a database field name.

eoDotAsColon - if this is True (by default), report generator will use "." as ":" in field names. For example - \a.field1\ will be interpreted as \a:field1\. This option appeared with v. 1.6. You may set this option to False if you need compatibility with first versions of EK RTF. Use scan blocks in new reports instead of fields with "." in MS Word table if you need to make cycle on a dataset.

eoNumericFormatClearZero - if this is True, all fields formatted with numeric formats such as fnum(), fcur() and so on, will be filtered for non-zero values. Zero numbers will be deleted from output result.

See also Format graphics

TEkRtf OnScanBefore event

Properties Methods Events Report template Example

```
type TEkOnScanBefore = procedure(ScanInfo:TEkScanInfo) of object;

property OnScanBefore : TEkOnScanBefore;
```

Event OnScanBefore appears before moving on first record of dataset when processing Scan-endscan block. Use this event with ScanInfo parameter to initialize variables and to make other necessary preparations for scan.

```
procedure TForm1.InvScanScanBefore(ScanInfo: TEkScanInfo);
begin
  case ScanInfo.Number of
      1: // before first scan in report template
      begin
         //Use Rows selected for the report in the grid
         ScanInfo.UseSelectedRows:=true;
         ScanInfo.SelectedRows:=Form2.DBGrid.SelectedRows;
         Total:=0;
      end;
      2: //before second scan in report template
      begin
      {.......}
      end;
   end;
end;
```

See also OnScanRecord, OnScanEof

TEkScanInfo
TEkScanInfo Properties

Unit
EkRtf

Description
Use TEkScanInfo to access Scan-endscan information in report.

TEkScanInfo Properties

See also OnScanBefore, OnScanRecord, OnScanEof events

TEkScanInfo properties
TEkScanInfo

property `Number:integer;`
Number of Scan in report template. For example:

```
\scan(Items)\         <-- 1
     \scan(Orders)\   <-- 2
     \endscan\

     \scan(Clients)\  <-- 3
     \endscan\
\endscan\
```

property `DataSet:TDataSet;`
DataSet used in Scan-endscan for moving.

property `SelectedRows:TBookMarkList;`
Scan moves from first to last record on dataset by default. You may specify TBookMarkList instance to scan dataset only on selected records.
For example:
`ScanInfo.SelectedRows := DBGrid1.SelectedRows.`
Property **UseSelectedRows** must be set to true.

property `UseSelectedRows:boolean;`
If this property=**true** then Scan uses property **SelectedRows** for moving on DataSet.

See also OnScanBefore, OnScanRecord, OnScanEof events

TEkRtf Execute method

```
procedure Execute(DS:Array of TDataSet);
```

This method reads input file, specified in property InFile (or specified with method SetTemplateBuffer), and puts result document into file specified in property OutFile.

See also ExecuteOpen, ExecuteStream

TEkRtf SetTemplateBuffer method

```
procedure SetTemplateBuffer(Buffer:pointer; Size:longint);
```

Sets pointer to memory area where program will search RTF template for report processing. Use this method instead of property `InFile`. For example, you may load report from blob field:

```
procedure TForm1.LoadTemplate;
var BS:TBlobStream;
    buffer:pointer;
    size:longint;
begin
 BS:=TBlobStream.create(Table1.FieldByName('BlobField') As
TBlobField ,bmRead);
 size:=BS.Seek(0,soFromEnd);
 BS.Seek(0,soFromBeginning);
 GetMem(Buffer,size);
 BS.Read((Buffer)^,size);
 BS.Free;
 EKRTF1.SetTemplateBuffer(Buffer, Size);
end;
```

See also FreeTemplate

TEkRTF methods
ClearVars
CreateVar
Execute
ExecuteOpen
ExecuteStream
FreeVar
SetTemplateBuffer
ShellOpenFile
txt2rtf
FreeTemplate
VarByName
Version

TEkRtf FreeTemplate method

Properties   Methods   Events   Report template

```
procedure FreeTemplate;
```

Frees memory previously associated with SetTemplateBuffer method.

TEkRtf ExecuteStream method

```
procedure ExecuteStream(DS:Array of TDataSet; OutStream:TStream);
```

This method reads input template and puts result document in stream specified in `OutStream` variable. You may use streams with TRichEdit or other third-party RTF control. For, example:

```
procedure TForm1.Button1Click(Sender: TObject);
var S:TMemoryStream;
begin
S:=TMemoryStream.Create;
Ekrtf1.ExecuteStream([],S);
RichEdit1.Lines.LoadFromStream(S);
S.Free;
end;
```

See also Execute, ExecuteOpen

TEkRtf ExecuteOpen method

```
procedure ExecuteOpen(DS:Array of TDataSet; ShowCmd:Integer);
```

This method is similar to `Execute`. After processing report template it runs output file with windows application associated with RTF files.
Additional parameter ShowCmd indicates how the application is to be shown when it is opened. This parameter can be one of the following values:

Value **Meaning**

SW_SHOW - Activates the window and displays it in its current size and position.

SW_MAXIMIZE - Maximizes the specified window.

SW_MINIMIZE - Minimizes the specified window and activates the next top-level window in the Z order.

SW_RESTORE - Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.

SW_SHOWDEFAULT - Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application. An application should call ShowWindow with this flag to set the initial show state of its main window.

SW_SHOWMAXIMIZED - Activates the window and displays it as a maximized window.

SW_SHOWMINIMIZED - Activates the window and displays it as a minimized window.

SW_SHOWNORMAL - Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

More details in Win32 API help.

See also Execute, ExecuteStream, ShellOpenFile

TEkRtf Format hyperlinks

You may insert values of variables and database fields as hyperlink. For this use function **flnk**. For example: **\flnk(var1)**. Expression for **var1** should be like **var1=http://www.yahoo.com** or **var1=c:\docs\report1.doc**. You may specify text name for the link using symbol "|", for example: **var1=c:\docs\report1.doc|The last report** or **var1=http://www.torry.net|Torry Delphi pages**.

Report generator does not change text attributes for hyperlink, so you may set it by yourself in report template: flnk(var1) .

TEkRtf OnScanRecord event

```
type TEkOnScanRecord = procedure(ScanInfo:TEkScanInfo) of object;

property OnScanRecord : TEkOnScanRecord;
```

Event OnScanRecord appears when dataset stays on a record in Scan-endscan block. You may use this event with ScanInfo parameter to calculate some variables and for other additional operations.

```
procedure TForm1.InvScanScanRecord(ScanInfo: TEkScanInfo);
begin
  case ScanInfo.Number of
      1: // first scan in report template
      begin
        Total:=Total+Table1.FieldByName('Sum').AsFloat;
      end;
      2: //second (nested) scan in report template
      begin
      {.......}
      end;
   end;
end;
```

See also OnScanBefore, OnScanEof

TEkRtf ExecuteSuccessful property

Methods Events Properties

`property ExecuteSuccessful:boolean;`

Read only. `True` if last called Execute method was performed successfully. `False` if errors (exceptions) were occured during report execution. Example:

```
try
EKRTF1.Execute([]);
except
{handling exceptions}
end;

if EKRTF1.ExecuteSuccessful then
{do something}
else showmessage('Can''t create report. Reason:'+EKRTF1.LastErrMsg);
```

See also: LastErrMsg

Scan functions

Properties Methods Events

You may add optional functions to commands "scan", "scanentry", "scanfooter", "endscan".
These functions are **SUM**, **CTN**, **CTS**. You may use these functions to sum or count values of data fields and report variables. Common format is:

```
\scancommand, ...., function1(source,destination)...functionN(...)\
```

Each function is performed when report generator gets corresponded scan command.

The first **argument** in each function is source data field or report variable. **Result** of each function is stored in report variable, that you specify as "destination" argument. Function may have **noreset** option - in this case its result will not be initiated with zero value if function was computed at least once. If result variable does not exists it will be created automatically.

Besides SUM, CTN, CTS functions you may call **user defined functions**. User defined functions are created through the UDFList property.

Sometimes it is necessary to declare report variable inside the report template, especially if this variable is an argument for user defined function. For this use **VAR** function.

See also creating report template, format functions

Scan functions - Cts

Properties Methods Events

CTS(SOURCE, DESTINATION [, NORESET]) - counts data field or report variable for not empty string values. String values containing only spaces are considered as empty.

source - data field or report variable to count.
destination - report variable to store result of the function.
noreset - use this option if you don't want to initialize destination variable with zero value. New not zero values of the source field will be counted beginning from the previous result of the function.

Example of a report:

```
\scan(a)\
\a:number\ - \a:svalue\
\endscan, ctn(a:svalue,c_value)\
Count of not empty string values: \c_value\

\scan(b)\
\b:number\ - \b:svalue\
\endscan, ctn(b:svalue,c_value,noreset)\
Count of not empty string values in both tables: \c_value\
```

Result may be like this:

```
1 - apples
2 -
3 - bananas
Count of not empty string values: 2

1 - tomatoes
Count of not empty string values in both tables: 3
```

See also scan functions

Scan functions - Sum

Properties Methods Events

SUM(SOURCE, DESTINATION [, NORESET]) - totals data field or report variable.

source - data field or report variable to sum.
destination - report variable to store result of the function.
noreset - use this option if you don't want to initialize destination variable with zero value. New values of the source field will be added to the previous result of the function.

Example of a report:

```
\scan(a)\
\a:number\ - \a:value\
\endscan, sum(a:value,s_value)\
total: \s_value\

\scan(b)\
\b:number\ - \b:value\
\endscan, sum(b:value,s_value,noreset)\
All totals: \s_value\
```

Result may be like this:

```
1 - 5
2 - 10
3 - 4
total: 19

1 - 10
All totals: 29
```

See also scan functions

Scan functions - Ctn

Properties   Methods   Events

CTN(SOURCE, DESTINATION [, NORESET]) - counts data field or report variable for values <> 0

source - data field or report variable to count.
destination - report variable to store result of the function.
noreset - use this option if you don't want to initialize destination variable with zero value. New not zero values of the source field will be counted beginning from the previous result of the function.

Example of a report:

```
\scan(a)\
\a:number\ - \a:value\
\endscan, ctn(a:value,c_value)\
Count of non zero values: \c_value\

\scan(b)\
\b:number\ - \b:value\
\endscan, ctn(b:value,c_value,noreset)\
Count of non zero values in both tables: \c_value\
```

Result may be like this:

```
1 - 5
2 - 0
3 - 4
Count of non zero values: 2

1 - 10
Count of non zero values in both tables: 3
```

See also scan functions

TEkRtf LastErrMsg property

Methods   Events   Properties

```
property LastErrMsg:string;
```

Read only. This property contains last error message, if errors (exceptions) were occured during last report execution. Use it together with ExecuteSuccessful property. Example:

```
if EKRTF1.ExecuteSuccessful then
{do something}
else showmessage('Can''t create report. Reason:'+EKRTF1.LastErrMsg);
```

See also: ExecuteSuccessful

TEkRtf DecimalRSeparator property

Methods Events Properties

`property DecimalRSeparator:char;`

Value of this property will be used in **ffixr** and **fnumr** format functions for float numbers. Char defined as DecimalRSeparator will appear instead of standard decimal separator in float numbers. This property is "-" by default.

For example, if DecimalRSeparator**='-'** and data field **Query1.field1=10.5** then field \ **fnumr(Query1:field1)** in report will be **10-5**

If fractional part of number equals 0 then DecimalRTerminator will be used.

See also DecimalRTerminator, Format functions

TEkRtf Format numbers

You may format values of numeric fields and variables using exponent, fixed, numeric or currency format. For this in report template use functions **fexp**, **ffix**, **ffixr**, **fnum**, **fnumr** or **fcur**. For example: **\fnum(a:field1)**, **\fcur(a:field2,4)**, **\fexp(a:field3,10:2)**

General format for all functions is **\func(name[,[precision:]decimals])**

Where **Name** specifies the name of data field or variable.

The **Precision** parameter specifies the precision of the given value. It should be 7 or less for values of type Single, 15 or less for values of type Double, and 18 or less for values of type Extended. Precision parameter may be omitted. By default Precision=18.

The meaning of the **Decimals** parameter depends on the particular function used. This parameter also may be omitted. By default Decimals=2 for **fexp**, **ffix**, **fnum** functions. Decimals=value of global variable **CurrencyDecimals** (unit SysUtils) when used **fcur**.

fexp(name[,[precision:]decimals])
Scientific format. The value is converted to a string of the form "-d.ddd...E+dddd". The resulting string starts with a minus sign if the number is negative, and one digit always precedes the decimal point. The total number of digits in the resulting string (including the one before the decimal point) is given by the Precision parameter. The "E" exponent character in the resulting string is always followed by a plus or minus sign and up to four digits. The Decimals parameter specifies the minimum number of digits in the exponent (between 0 and 4).

ffix(name[,[precision:]decimals])
Fixed point format. The value is converted to a string of the form "-ddd.ddd...". The resulting string starts with a minus sign if the number is negative, and at least one digit always precedes the decimal point. The number of digits after the decimal point is given by the Decimals parameter--it must be between 0 and 18. If the number of digits to the left of the decimal point is greater than the specified precision, the resulting value will use scientific format.

ffixr(name[,[precision:]decimals])
The same as **ffix** function, except that DecimalRSeparator property will be used instead of default decimal separator char. DecimalRTerminator char will be at the end of integer values.

fnum(name[,[precision:]decimals])
Number format. The value is converted to a string of the form "-d,ddd,ddd.ddd...". The fNum function corresponds to the fFix function, except that the resulting string contains thousand separators.

fnumr(name[,[precision:]decimals])
The same as **fnum** function, except that DecimalRSeparator property will be used instead of default decimal separator char. DecimalRTerminator char will be at the end of integer values.

fcur(name[,[precision:]decimals])
Currency format. The value is converted to a string that represents a currency amount. The conversion is controlled by the CurrencyString, CurrencyFormat, NegCurrFormat, ThousandSeparator, and DecimalSeparator global variables, all of which are initialized from the Currency Format in the International section of the Windows Control Panels. The number of digits after the decimal point is given by the Decimals parameter--it must be between 0 and 18.

See also creating report template, Format functions

TEkRtf DecimalRTerminator property

Methods Events Properties

property `DecimalRTerminator:char;`
Value of this property will be used in **ffixr** and **fnumr** format functions for float and integer numbers. Char defined as DecimalRTerminator will appear at the end of integer or float number without fractional part.
This property is "=" by default.
For example, if DecimalRTerminator**='='** and data field **Query1.field1=10.0** then field **\fnumr(Query1:field1)\** in report will be **10=**

See also DecimalRSeparator, Format functions

TEkRtf OnScanEof event

Properties   Methods   Events   Report template   Example

```
type TEkOnScanEof = procedure(ScanInfo:TEkScanInfo) of object;

property OnScanEof : TEkOnScanEof;
```

Event OnScanEof appears after moving from last record of dataset in Scan-endscan block. You may use this event with ScanInfo parameter to make some additional operations.

```
procedure TForm1.InvScanScanEof(ScanInfo: TEkScanInfo);
begin
  case ScanInfo.Number of
      1: // first scan in report template
      begin
        EkRTF1.Add('Total='+FloatToStr(Total));
      end;
      2: //second (nested) scan in report template
      begin
      {.......}
      end;
   end;
end;
```

See also OnScanBefore, OnScanRecord

TEkRtf Format dates

Function **fdtm** uses **FormatDateTime** Delphi function to format dates. Syntax is **fdtm(dataset:field,format_var)** or **fdtm(variable,format_string)**, where **format_string** is string constant or report variable with format pattern for FormatDateTime function.

For example:
var1='2/15/95 10:30am'
format_var='"The meeting is on" dddd, mmmm d, yyyy, ' + '"at" hh:mm AM/PM'

field **\fdtm(var1,format_var)** in report will be: The meeting is on Wednesday, February 15, 1995 at 10:30 AM

See help on FormatDateTime function for details.

TEkRtf CreateVar method

```
procedure CreateVar(Name:string; Value:string);overload;
procedure CreateVar(Name:string; Value:Double);overload;
procedure CreateVar(Name:string; Value:TDateTime;
IgnoreTime:boolean);overload;
procedure CreateVar(Name:string; Value:boolean);overload;
```

This method creates report variable using the Value of certain type and inserts it in the VarList.

String variables are stored in VarList without any conversions:
`EKRTF1.CreateVar('Var1','Text line 1');`
In VarList you'll have a string `'Var1=Text line 1'`

Float variables are stored in VarList by FloatToStr function. Example:
`EKRTF1.CreateVar('Var1',10.5);`

TDateTime variables are stored by functions DateToStr (without time) or DateTimeToStr (with time).
To specify whether you want to store time part or not, use **IgnoreTime** variable. For example, if you are using an expression like `EKRTF1.CreateVar('TodayDate',Now(),true),` you'll have value of **TodayDate** like **10/05/2001**.

If you add this variable using `EKRTF1.CreateVar('TodayDate',Now(),false),` you'll have value of **TodayDate** something like **10/05/2001 9:15:42**.

Boolean variables are stored in VarList according with TrueValue and FalseValue properties. For example, if **TrueValue**='Yes' and **FalseValue**='No', you'll have strings in VarList `'Var1=Yes'` for True values, and `'Var1=No'` for False values.
If for some reason you set property TrueValue equal to FalseValue, CreateVar with boolean variable will use strings 'True' and 'False' to add it to VarList.

If report variable with given Name already exists, you'll get an exception with message 'Can't add report variable ...'.

To manipulate with existing report variables in code use function VarByName.

See also FreeVar, ClearVars

TEkRtf ClearVars method

Properties   Methods   Events

```
procedure ClearVars;
```

Deletes all report variables. Example:
`EKRTF1.ClearVars;`
Identical to `EKRTF1.VarList.clear;`

See also FreeVar, CreateVar

TEkReportVariable
TEkReportVariable Properties

Unit
EkRtf

Description
TEkReportVariable is used to manipulate with report variables as with integer, float, date, string or boolean values.
See TEkReportVariable properties for details.

See also TEkRTF VarByName method

f(x) TEkUDFList component

TEkUDFList Properties————TEkUDFList Methods

Unit
EkFunc

Description

TEkUDFList is class derived from TComponent. It is used in conjunction with TEkRTF to centralize the response to user defined functions in the RTF report template. UDF list component contains the Functions property, which is collection of TEkUDF items . Add TEkUDFList component to your form or data module, open the Functions property to display the UDF list editor, from which you can add, delete, and rearrange functions. Each TEkUDF item has OnCalculate event where you can write code of user defined function.

TEkReportVariable AsInteger property

TEkReportVariable TEkReportVariable Properties

Represents the value of the report variable as an integer value.

```
property AsInteger: Int64;
```

Reading AsInteger converts the value of the report variable to an integer using the StrToInt64 function. Setting AsInteger converts the integer to a string using the IntToStr function.

TEkUDFList properties

TEkUDFList TEkUDFList methods

Count
Functions

TEkReportVariable AsBoolean property

TEkReportVariable   TEkReportVariable Properties

Represents the value of the report variable as a boolean value.

```
property AsBoolean: Boolean;
```

AsBoolean returns True on reading the value of the report variable if its text is the same as TrueValue string or if its text is word 'True' (not case sensitive). False returned in all other conditions.
Setting AsBoolean converts boolean value to a string using TrueValue or FalseValue properties. If for some reason you set property TrueValue equal to FalseValue, AsBoolean will use strings 'True' and 'False' to store the value of the variable.

TEkReportVariable properties

TEkReportVariable

AsString
AsFloat
AsInteger
AsDate
AsDateTime
AsBoolean
Name

TEkReportVariable AsFloat property

TEkReportVariable TEkReportVariable Properties

Represents the value of the report variable as a floating-point value.

```
property AsFloat: Double;
```

Reading AsFloat converts the value of the report variable to a floating-point value using the StrToFloat function. Setting AsFloat converts the floating-point value to a report variable using the FloatToStr function.

TEkReportVariable AsString property

TEkReportVariable   TEkReportVariable Properties

Represents the value of the report variable.

```
property AsString: string;
```

This property provides a uniform interface that allows applications to get or set string values of report variable. String values need no conversion, because the native format of a report variable in VarList is a string.

TEkReportVariable AsDate property

TEkReportVariable   TEkReportVariable Properties

Represents the value of the report variable as value of date.

```
property AsDate: TDateTime;
```

Reading AsDate converts the value of the report variable to a date-time using the StrToDate function.
Setting AsDate converts the date-time value to a string using the DateToStr function.

See also AsDateTime

TEkRtf FreeVar method

Properties   Methods   Events

```
procedure FreeVar(Name:string);
```

Deletes report variable identified by Name. Example:

```
EKRTF1.FreeVar('Var1');
```

See also ClearVars, CreateVar

TEkReportVariable AsDateTime property

TEkReportVariable   TEkReportVariable Properties

Represents the value of the report variable as TDateTime value.

```
property AsDateTime: TDateTime;
```

Reading AsDateTime converts the value of the report variable to a date-time using the StrToDateTime function. Setting AsDateTime converts the date-time value to a string using the DateTimeToStr function.

See also AsDate

TEkReportVariable Name property

TEkReportVariable   TEkReportVariable Properties

```
property Name: String;
```

Returns the name of the variable wich was used in method CreateVar of TEkRTF.

TEkUDFList methods

TEkUDFList   TEkUDFList properties

Create
Destroy
FindFunction
Version

TEkUDFList Count property

TEkUDFList　　TEkUDFList properties　　TEkUDFList methods

```
property Count:integer;
```

Returns the count of user defined functions in collection Functions of TEkUDFList component.

TEkUDFList Create method

TEkUDFList TEkUDFList properties TEkUDFList methods

```
constructor Create(AOwner: TComponent);override;
```

Creates an instance of a TEkUDFList component.

Call Create to instantiate an UDF List declared in an application if it was not placed on a form at design time. Create calls its inherited Create constructor and creates an empty Functions collection.

TEkUDFCollection

Unit
EkFunc

Description

TEkUDFCollection is class derived from TCollection. It holds the collection of TEkUDF objects.

Methods:

```
constructor Create(FnList:TEkUDFList);
```

Constructor creates new TEkUDFCollection object. FnList is TEkUDFList component which holds the TEkUDFCollection object. You don't need to call it directly if you place TEkUDFList component on a form or data module at design time.

```
function Add:TEkUDF;
```

This function adds new TEkUDF objects to the collection. At design time you may use UDF List Functions property editor to add UDF to the collection.

Properties:

```
property Items[Index:integer]:TEkUDF;
```

Use Items to access individual TEkUDF object in the collection. The value of the Index parameter corresponds to the Index property of TEkUDF item. It represents the position of the item in the collection.

See also: TEkUDFList, TEkUDF

TEkUDFList Destroy method

TEkUDFList TEkUDFList properties TEkUDFList methods

```
destructor Destroy;override;
```

Do not call Destroy directly. Instead call Free to verify that the component is not already freed before calling Destroy. Destroy frees the Functions collection of UDF List, and then calls its inherited Destroy destructor.

TEkUDFList FindFunction method

TEkUDFList TEkUDFList properties TEkUDFList methods

```
function FindFunction(Name:string):integer;
```

If UDF with specified name exists in the Functions collection, FindFunction returns its number from 0 to Functions.Count-1. If UDF with such name doesn't exists FindFunction returns -1.

TEkUDFList Functions property

TEkUDFList   TEkUDFList properties   TEkUDFList methods

```
property Functions:TEkUDFCollection;
```

Returns the collection of user defined functions in TEkUDFList component.

TEkRtf VarByName method

Properties   Methods   Events

```
function VarByName(VarName:string):TEkReportVariable;
```

Returns report variable specified in VarName string as TEkReportVariable object. Example:

```
n:=EKRTF1.VarByName('Var1').AsInteger;
EKRTF1.VarByName('DateVar').AsDate:=Now();
```

Variable must be already defined when you use it with VarByName function.

See also CreateVar

TEkUDF

TEkUDF properties TEkUDF events

Unit
EkFunc

Description

TEkUDF is class derived from TCollectionItem. It is used in conjunction with TEkUDFCollection object. It represents the definition of user function defined in RTF report template. RTF report template is InFile in TEkRTF component.

See also: TEkUDFList

TEkRtf UDFList property

```
property UDFList:TEkUDFList;
```

Specifies the UDF List component with user defined functions.

Scan functions - Var

Properties   Methods   Events

VAR(VAR1, ... , VAR N) - creates variables VAR1, ... VAR N if they don't exist.

Initial value for each new created variable is empty string. If report variable exists, it is not changed in any way. When you use declared variable with UDF, you may need to init its value by your own code.

Example:

```
\scan(a), var(totsal), myinit(totsal)\
-----------------------------
\scan(b), mysum(b:field1,totsal)\
\b:number\    \b:field1\
\endscan\
total: \totsal\
\endscan\
```

See also scan functions, constants

TEkRtf ShellOpenFile method

```
function ShellOpenFile(const FileName:string; ShowCmd: Integer): THandle;
virtual;
```

This function opens file specified in FileName using associated windows application.
If the function succeeds, the return value is the instance handle of the application that was run. If the function fails, the return value is an error value that is less than or equal to 32.

Additional parameter ShowCmd indicates how the application will be shown after opening. It works the same way as in ExecuteOpen method.

More details in Win32 API help.

TEkRtf txt2rtf method

Properties   Methods   Events

```
function txt2rtf(s:string):string;
```

Converts text string s into its RTF code representation. This function is used internally. You don't need to call it directly from your application.

InsertRtfMemo procedure

Inserts rich formatted text in the report.

Unit
EkRtfStream

```
procedure InsertRtfMemo(Sender:TObject; OutputStream:TStream; var
RtfContent:string);
```

InsertRtfMemo procedure may be called from a user defined function with result type **udfrTMemoryStream**.
Sender is a parameter of TEkRtf type. **OutputStream** is an output report results stream, usually it is UDFResult object in the UDF OnCalculate event code. **RtfContent** is RTF formatted string copied from a rich text blob field or rtf file.

When using InsertRtfMemo within UDF, always set the ResultType of user function to **udfrTMemoryStream**.

There is an example code of user function named "InsertRtf". Function has one argument - the name of blob data field.

```
procedure TForm1.EkUDFList1Functions0Calculate(Sender: TObject;
  Args: TEkUDFArgs; ArgCount: Integer; UDFResult: TObject);
var s:string;
begin //************** code of user function InsertRtf(DataField) *****
  if not (UDFResult is TMemoryStream) then raise Exception.Create('Result type
of user function InsertRtf must be set to udfrTMemoryStream!');

  If not (Args[0] is TBlobField) then raise Exception.Create('Blob data field
as argument expected in user function InsertRtf!');

  s:=TBlobField(Args[0]).AsString;

  InsertRtfMemo(Sender, TMemoryStream(UDFResult), s);
end;
```

In a report template reference to this function looks like **\InsertRtf(Table1:RichTextField)**

See also Insert picture example, InsertRtfMemoStream

TEkUDF properties, events

TEkUDF

Properties:

ArgMinCount
ArgMaxCount
Name
ResultType

Events:

OnCalculate

TEkUDF Name property

TEkUDF TEkUDF properties

```
type TEkUDFName=string;
property Name:TEkUDFName;
```

Use this property to set the name of the user defined function which you use in the RTF report template. Name of the function is case insensitive. Rules for function naming is the same as for identifiers.

See also: TEkUDFList

TEkUDF ArgMinCount property

TEkUDF TEkUDF properties

```
property ArgMinCount:word;
```

Use this property to set minimal required number of arguments for user defined function. If number of arguments is less than ArgMinCount, exception EIllegalFunctionUse will be generated.

See also: ArgMaxCount, TEkUDFList

TEkRtf Version method

```
function Version:longint;
```

Returns number of EK RTF component version as integer value. For example, version 1.81 will return number 181 and so on ...

TEkUDF ArgMaxCount property

TEkUDF TEkUDF properties

```
property ArgMaxCount:word;
```

Use this property to set maximum allowed number of arguments for user defined function. If number of arguments is more than ArgMaxCount, exception EIllegalFunctionUse will be generated.

See also: ArgMinCount, TEkUDFList

TEkUDF ResultType property

TEkUDF TEkUDF properties

```
type TEkUDFResultType=0..255;
property ResultType:TEkUDFResultType;
```

Use this property to define what type of result will process the user defined function. Type of result is the actual type of UDFResult TObject variable in the event OnCalculate.

There are constants defined in the unit ConsCom for ResultType property:

```
const udfrNil=0;
      udfrTEkReportVariable=1;
      udfrTPicture=2;
      udfrTMemoryStream=3;
```

Your code must operate with UDFResult object in according with its ResultType property. If ResultType<>**udfrNil**, UDFResult object will be created before processing OnCalculate event.

Example:
Let ResultType=udfrTEkReportVariable. In the code you may operate with result as shown below

```
with UDFResult as TEkReportVariable do begin
     AsString:='abc';
end;
```

or

```
TEkReportVariable(UDFResult).AsString:='abc';
```

udfrTMemoryStream is used internally. Don't use it unless your want to write directly into output RTF code.

See also: TEkUDF OnCalculate event, TEkUDFList

InsertRtfMemoStream procedure

Inserts rich formatted text in the report.

Unit
EkRtfStream

```
procedure InsertRtfMemoStream(Sender:TObject; OutputStream:TStream; var
RtfContent:TStream);
```

InsertRtfMemoStream procedure may be called from a user defined function with result type **udfrTMemoryStream**.
Sender is a parameter of TEkRtf type. **OutputStream** is an output report results stream, usually it is UDFResult object in the UDF OnCalculate event code. **RtfContent** is stream with RTF formatted text copied from a rich text blob field, rtf file or from another stream.

When using InsertRtfMemoStream within UDF, always set the ResultType of user function to **udfrTMemoryStream**.

There is an example code of user function named "InsertRtf". Function has one argument - the name of blob data field.

```
procedure TForm1.EkUDFList1Functions0Calculate(Sender: TObject;
  Args: TEkUDFArgs; ArgCount: Integer; UDFResult: TObject);
var st:TStream;
begin //************** code of user function InsertRtf(DataField) *****
  if not (UDFResult is TMemoryStream) then raise Exception.Create('Result type
of user function InsertRtf must be set to udfrTMemoryStream!');

  If not (Args[0] is TBlobField) then raise Exception.Create('Blob data field
as argument expected in user function InsertRtf!');

  st:=TStringStream.Create(TBlobField(Args[0]).AsString);

  InsertRtfMemoStream(Sender, TMemoryStream(UDFResult), st);
  st.Free;
end;
```

In a report template reference to this function looks like **\InsertRtf(Table1:RichTextField)**

See also Insert picture example, IsertRtfMemo

TEkUDF OnCalculate event

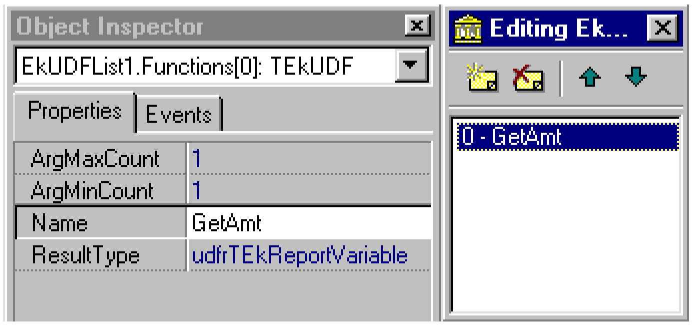
TEkUDF TEkUDF properties

```
type
TEkUDFArgs=array of TObject;
TEkUDFResult=TObject;

TEkUDFOnCalculate = procedure(Sender:TObject; Args:TEkUDFArgs;
ArgCount:integer; UDFResult:TEkUDFResult) of object;

property OnCalculate:TEkUDFOnCalculate;
```

Use this event to realize code responsible for UDF action.
The **Sender** parameter in an event handler informs Delphi which component received the event, and therefore called the handler.
Args is array of TObjects passed to function. It may be TEkReportVariable, TField or result of another UDF function. **Constants** passed to a function become in **Args** as report variables.
ArgCount is number of arguments.
Result of user function is **UDFResult** object. Type of UDFResult defined in ResultType property.

Example:

This is definition of user function with name GetAmt with one numeric argument. Argument may be database field or report variable. Function returns string 'Zero' if argument=0, otherwise function returns argument itself as string.



EkUDFList1.Functions[0]: TEkUDF

ArgMaxCount	1
ArgMinCount	1
Name	GetAmt
ResultType	udfrTEkReportVariable

0 - GetAmt

```
procedure TForm1.EkUDFList1Functions0Calculate(Sender: TObject;
  Args: TEkUDFArgs; ArgCount: Integer; UDFResult: TObject);
var s:string;
    n:Double;
begin
//************ GetAmt UDF Code ***********************

  s:='';

  if Args[0] is TField then s:=TField(Args[0]).AsString;
  if Args[0] is TEkReportVariable then s:=TEkReportVariable(Args[0]).AsString;

  With UDFResult as TEkReportVariable do begin
     n:=StrToFloat(s);
     if n=0 then AsString:='Zero' else AsString:=s;
  end;
```

```
end;
```

In a report template you may reference to the user function as **\GetAmt(A:AmountPay)** or **\GetAmt(var1)**.

See also: user defined function TPicture example

TEkUDF OnCalculate event TPicture example

TEkUDF TEkUDF properties

This is an example of user defined function with ResultType **udfrTPicture**. This function has no any arguments, it just gets a **TImage** component Image1, which is on a Form1, draws its picture.graphic on a result bitmap and returns it to the calling TEkRTF component.

```
procedure TForm1.EkUDFList1_GetPicture_Calculate(Sender: TObject;  Args:
TEkUDFArgs; ArgCount: Integer; UDFResult: TObject);
begin
   With UDFResult as TPicture do begin
       Bitmap.Width:=Image1.Width;
       Bitmap.Height:=Image1.Height;
       Bitmap.Canvas.Draw(0,0,Image1.Picture.Graphic);
   end;
end;
```

See also: OnCalculate event, InsertRtfMemo

TEkUDFList Version method

TEkUDFList   TEkUDFList properties   TEkUDFList methods

```
function Version:longint;
```

Returns version number of   UDF List component as integer value. For example, version 1.70 will return number 170 and so on ...