TMS TAdvEdit
DEVELOPERS GUIDE

Jul 2016

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

Index

Availability

TMS TAdvEdit is available for Delphi 7, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, C++Builder 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle (Prof/Enterprise/Architect)

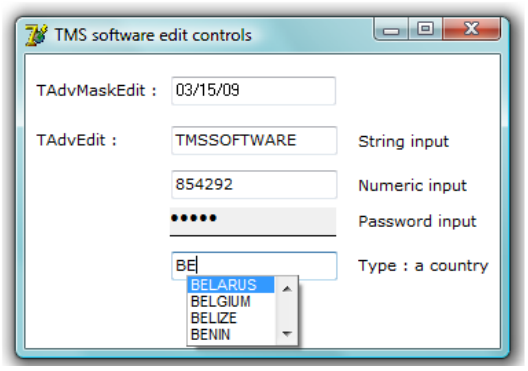
TMS TAdvEdit has been designed for and tested with: Windows Vista, Windows 7, Windows 8, Windows 10.

Online references

TMS software website:
http://www.tmssoftware.com

TMS HTML Controls Pack page:
http://www.tmssoftware.com/site/advedit.asp

TAdvEdit



TAdvEdit description

Edit controls with lots of extra capabilities.

TAdvEdit features

- data-aware & non data-aware version of advanced edit control and advanced mask edit control
- focus color, focus border color, colored on modify, disabled color
- move focus to next control with return or tab
- numeric right aligned input
- uppercase, lowercase, mixed case input
- numeric & alpha numeric only input
- auto focus, focus border
- attached label
- flat style
- fixed suffixes and prefixes
- different types : numeric, float, money, hex, range
- precision for float edit style
- automatic thousand separator inserting for money style input
- URL aware
- associated AdvInputQuery method that improves the standard InputQuery with the many TAdvEdit functions.
- properties to obtain the value as integer or float for numeric and float edit style
- events for all clipboard actions
- automatic persistence through INI file or registry
- transparency
- OLE drag and drop support
- different color when edit value is invalid
- different width & alignment during focus
- empty text string to display grayed text while edit control is empty
- automatic thousand separator insertion
- Excel style decimal separator behaviour
- Error marking while not editing
- Outlook style lookup popup with various settings to control look and behaviour

TAdvEdit use

TAdvEdit is an enhanced TEdit control that has many options to make the data entry process much more user friendly. One of its capabilities is to do a validation while typing of several formats. TAdvEdit descends from TEdit, so all standard VCL functionality of the TEdit control remains available.

Behaviour extensions of TAdvEdit

The main property to control what data can be entered in the edit control is the EditType property. Following types are avaiable:

etString: just like the TEdit any text entry is possible
etNumeric: only numeric data entry is possible. No decimal or thousand separator can be entered. The Signed property controls whether the minus signed can be entered or not.
etFloat: floating point data can be entered, this is numbers, a thousand and/or decimal separator
etUppercase: characters are automatically converted to uppercase during entry
etMixedCase: first character of a word is automatically converted to uppercase
etLowerCase: characters are automatically converted to lowercase during entry
etPassword: entered characters are not displayed but an asterix is shown inside. This character is set with the PasswordChar property
etMoney: numbers, thousand & decimal separator and currency symbol can be entered
etRange: a range of values can be entered, separated by a semi colon.
etHex: hex values can be entered, ie. numbers 0 to 9 and characters A..F
etAlphaNumeric: Only alphabetic and numbers 0 to 9 can be entered
etValidChars: only characters that are part of the ValidChars property can be entered

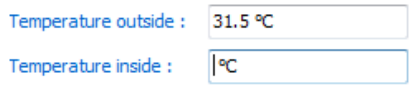
Other properties that control the behavior of the edit control are:

AllowNumericNullValue: boolean : when true, when no text is entered in the edit control, this behaves as if a null is entered. This is especially usefull in database applications where the numeric field value can be set as null.
AutoFocus: boolean : when true, the edit control gains focus automatically when the mouse hovers the edit control.
AutoSelect: boolean : when true, all text is selected when the focus moves to the edit control
AutoThousandSeparator: boolean : when the EditType is etMoney or etFloat, a thousand separator will be automatically inserted while typing.
ExcelStyleDecimalSeparator: boolean : when true, the decimal separator entered is the system defined decimal separator when '.' is pressed on the numerical keypad.
ReturnIsTab: boolean : when true, pressing return behaves like pressing a tab key.
TabOnFullLength: boolean : when true, when the number of characters in the edit control reaches MaxLength, focus will move automatically to the next control

A fixed suffix and/or prefix for the edit control

It can be convenient that there is a fixed suffix or prefix hinting the user typically of the dimension of a value to enter. This can be the currency symbol for entering monetary values or a dimension like degrees Celcius °C.

To have this dimension always displayed and as non editable part of the text, it can be set via Suffix and/or Prefix property.



Display capabilities of TAdvEdit

When the edit control has no text, the property EmptyText determines the text displayed in a light gray color in the edit control. This text typically serves as a hint for the user about what to enter. The property EditAlign controls the alignment of the text in the edit control. This can be particulary useful for numeric entry where it is often more convenient for the user to enter the text as right-aligned.

Another useful feature is to define a different colors for when the edit control has focus. This consists of:

FocusAlign: sets the text alignment when the control has focus
FocusBorder: boolean : when true, the border is painted in FocusBorderColor when the control has focus
FocusColor: TColor : sets the background color when the edit control has focus
FocusBorderColor: TColor : sets the color of the border when the edit control has focus
FocusFontColor: TColor : sets the font of the text color when the edit control has focus
FocusIncWidth: integer : when set to a value larger than 0, the control's width can increase when it gets focus

Other visual control is available with:

ShowModified: boolean : when true, the text is displayed in ModifiedColor color when it was changed by the user.
ModifiedColor: TColor : sets the color of text when it was modified by the user
ShowURL: boolean : when true, when the text entered is a hyperlink (ie. starts with http://, ftp:// etc...) the text is painted in the hyperlink color and is underlined.
URLColor: TColor : sets the color of the hyperlink when shown in the edit control

Label attached to the edit control

Optionally, a conveniently attached text label to the edit control can be used. Set the text of the label with Edit.LabelCaption. The label can be positioned at various positions with respect to the edit control. This is controlled by the LabelPosition property.

Additionally, the font of the label can be set with LabelFont, the margin between the label and edit control with the LabelMargin property and the label can be set transparent with LabelTransparent: boolean. When the property LabelAlwaysEnabled is set to true, the label is shown in enabled state

irrespective of the edit control enabled state. Otherwise, the enabled state of the label and edit control are equal and controlled by TAdvEdit.Enabled.

Built-in Lookup or autocompletion capabilities and persistence of TAdvEdit

Just like in Outlook when you start typing an email address, Outlook makes a suggestion while typing on possible matching email addresses, TAdvEdit has such capability. The settings to control this lookup or autocompletion capability can be found under TAdvEdit.Lookup:

CaseSensitive: boolean : when true, match for lookup is case sensitive
Color: TColor : sets the color of the lookup dropdown control
DisplayCount: integer : sets the number of items to show before a vertical scrollbar appears
DisplayList: TStringList : list of values to display in the dropdown lookup list
Enabled: boolean : when true, lookup or autocompletion functionality is enabled
History: boolean : when true, all entered values in the edit control are automatically remembered in the DisplayList
Multi: boolean : when true, lookup is performed for every part in the edit control separated by the Separator character
NumChars: integer : sets the number of characters that need to be entered in the edit control before lookup starts
Separator: character : sets the character that separates multiple text parts lookup can be performed for
ValueList: TStringList : list of values that will be used as lookup

The default and most simple behavior of the lookup is to add entries in the Edit.Lookup.DisplayList stringlist. For example:

```
begin
  with AdvEdit1.Lookup.DisplayList do
  begin
    Clear;
    Add('BMW');
    Add('Mercedes');
    Add('Porsche');
    Add('Ferrari');
    Add('Audi');
  end;
  AdvEdit1.Lookup.Enabled := true;
end;
```

This initializes the lookup list with some values and typing 'Fe' wil show the suggestion to complete to 'Ferrari'. When the ValueList is used, a different value can be entered in the edit control from what is displayed in the dropdown list. This could be used like in following example:

```
begin
  with AdvEdit1.Lookup do
  begin
    DisplayList.Clear;
    ValueList.Clear;
    DisplayList.Add('BMW');
    ValueList.Add('http://www.bmw.de');
    DisplayList.Add('Mercedes');
    ValueList.Add('http://www.mercedes.de');
    DisplayList.Add('Porsche');
```

```
    ValueList.Add('http://www.porsche.de');
    DisplayList.Add('Ferrari');
    ValueList.Add('http://www.ferrari.it');
    DisplayList.Add('Audi');
    ValueList.Add('http://www.audi.de');
  end;
  AdvEdit1.Lookup.Enabled := true;
end;
```

Before selection:

After selection:

This code snippet initializes the lookup with display text and value text. When typing 'Fe', the dropdown list will suggest 'Ferrari' and when accepting to autocomplete, the edit control text will be set to 'http://ferrari.it'

When Lookup.History is set to true, values entered in the edit control are automatically added to the DisplayList.

This DisplayList can be persisted in the registry or in an INI file. To enable this capability, set AdvEdit.Persistence.Enable is true and specify the registry key or INI filename via AdvEdit.Persistence.Key and the registry value name or INI file section name

with AdvEdit.Persistence.Section.

TAdvEdit tips and FAQ

Allowing to enter negative & positive numbers

Set AdvEdit.EditType = etNumeric;
Set AdvEdit.Signed = true;

Using the lookup list

Drop a default TAdvEdit on the form and initialize the lookup with code similar to:

```
with AdvEdit1.Lookup do
begin
  Enabled := true;
  DisplayList.Add('Audi');
  DisplayList.Add('BMW');
  DisplayList.Add('Bugatti');
  DisplayList.Add('Ferrari');
  DisplayList.Add('Ford');
  DisplayList.Add('Mercedes');
end;
```

type at least 2 characters (default setting) and the lookup list should display.

Creating a TAdvEdit at runtime

To initialize focus handling properly when creating a TAdvEdit at runtime, call TAdvEdit.Init after creation.

What is the DBAdvEdit.EmptyText property?

EmptyText is the text that is displayed in the control when the Text property of the edit control is an empty string. EmptyText as such is just a display value and is unrelated to what is persisted in the databse. EmptyText is typically a grey text suggesting what the user should enter and disappears as soon as the user typed text.

Caret not visible in the edit control

It is a known shortcoming of the Windows operating system EDIT class (that standard VCL TEdit and also TMS TAdvEdit uses) that the caret is not displaying when the height of the control is too small in relation to the choosen font. As this is a Windoiws limitation, it can only be solved by either increasing the control height or decreasing the font size.

Using the decimal separator when '.' is pressed on the numerical keypad

Set AdvEdit.ExcelStyleDecimalSeparator = true