Table of Contents

1 TAppBar - Overview

Overview

The AppBar component lets your forms to behave like an Application Desktop Toolbar — to dock on the edges of the screen like usual taskbar or MS-Office panel. Though appbars are usually docked on an edges of the user's screen, they also can float as usual windows.

When the AppBar is anchored to the screen edge, it can be automatically hidden from screen when other window activated and popup on screen again when user point the mouse to the thin line on screen edge. User can dock or undock the appbar just moving it with mouse pointer. All automatical movements of the appbar can be displayed with smooth sliding effect. Component contains a whole bunch of neat additional features, such like docking and sizing rules, float restrictions, registry saver, animation effects and so forth...

How to use ?

Most simple and quickest way to test the Application Desktop Toolbar features — drop AppBar component on your form, compile and execute your program. Then try to move your form driving it to the screen edges. Form will be docked and undocked by mouse pointer. You may change the Placement property in your program and docking placement will be changed accordingly with smooth animation effect (if Sliding is enabled).

You can also specify initial placement at design-time, changing value of Placement property. To disallow docking on some screen edges — use AllowedEdges propery. To let the AppBar automatically hide from screen — make AutoHide property True.

Also, the acAppBar can automatically save and restore form placement on every program restart, see RegistrySaver structure.

Basic features / properties

AllowedEdges	specifies whether your form is able or unable to be docked to enumerated edges;
AlwaysOnTop	ensures that the appbar is always visible, even when you run another programs in a maximized window;
AutoHide	controls whether AppBar can be hidden (reduced to a thin line) at the edge of screen;
Placement	controls current AppBar placement. Change the Placement property to dock / undock the AppBar to specified edge. The replacement can be shown with smooth animation effect if Sliding effect is enabled;
LastDockingPlace	determines the previous AppBar placement. When user doubleclicks the AppBar, it will restore previous placement;
Docking	structure controls the AppBar size when form is docked to the screen edge. You can enable or disable docking rules, specify maximum and minimum horizontal and vertical size when form is docked, specify width and height of the AppBar when it docked to the screen edge;
Sizing	structure controls sizing rules of the AppBar. You can specify the horizontal and vertical increment of the form with AppBar;
Sliding	structure controls whether you would like to show automatic AppBar movements with smooth sliding effect. In example, your form is docked to the screen edge and it able to auto-hide from screen. When user move mouse pointer to thin line on screen edge, for will appears with sliding effect. Time of effect can be specified

	in the SlideTime property;
FloatRestrictions	structure controls the size restrictions when form in normal, "floating" state (when Placement property is bpFloat);
RegistrySaver	saves and restores from the system registry all AppBar settings to the on program restart;
TaskIcon	specifies whether you would like to show the program icon on taskbar when the AppBar docked to screen edge.

Some frequently asked questions

Q. When user doubleclicks the form, the AppBar will restore previous placement [*LastDockingPlace property*]. Is there a way to avoid re-positioning of the AppBar ?
A. To disallow restoring of previous placement on doubleclick, just hook OnDblClick event.

Q. I have docked the AppBar to the left border of screen on program startup but the form appears with the same width as in "floating" state!
A. Seems that your form contains a lot of non-resizeable (not aligned) controls and should have enough space to show them, however can not auto-scroll for displaying. Just set "AutoScroll" property of your form to True.

Q. I have docked the form to the screen edge, but seems I can not resize it. Your demo programs allows to resize docked appbars. I have also checked Enabled property of Sizing structure and it's True. What I doing wrong?
A. Sure, you can not resize forms with fixed placement (i.e.: BorderStyle = bsDialog or bsSingle). Just make your form resizeable, set BorderStyle property to bsSizeable or bsSizeToolWin.

Q. Is there a way to delay the popup when anchored to the edge of the screen [*when the AppBar is "autohidden"*]? (I've had a couple users complain that they moved their mouse to fast when trying to get to a desktop icon or somewhere else and went to far and caused the appbar to pop out even though they quickly moved it away)
A. Yes. You should hook OnUnhiding event and unhide the AppBar with delay (delay can be implemented with TTimer component), using ShowHiddenAppBar method. Please check out an **example** for more details.

Q. I have a form with AppBar on it. There is a panel on the form. This hides part of the form. When I try to drag the form to dock it to the side of the screen or to undock it obviously now it doesn't work. Could you suggest a way in which I could fire the event which docks, undocks and moves the form when the user clicks or tries to drag the panel component.
A. In **v1.7** we have added bonus component, **TDragPanel**. You can drop this panel to the AppBar and user will be able to move and dock the appbar dragging this panel.

2 Installation Instruction

Package without source code

to Delphi 2

1. Unzip files from "Delphi2" directory to your "Delphi 2\Lib" directory.
2. Start Delphi 2 IDE.

3. Select "Component \ Install..." menu item.
4. Press "Add" button and select "_AppBarReg.pas" file.
5. Rebuild library.

to Delphi 3
1. Unzip files from "Delphi3" directory and copy them to "Delphi 3\Lib".
2. Start Delphi 3 IDE.
3. Open "AppBarD3.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 4
1. Unzip files from "Delphi4" directory and copy them to "Delphi 4\Lib".
2. Start Delphi 4 IDE.
3. Open "AppBarD4.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 5
1. Unzip files from "Delphi5" directory and copy them to "Delphi 5\Lib".
2. Start Delphi 5 IDE.
3. Open "AppBarD5.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 7
1. Unzip files from "Delphi7" directory and copy them to "Delphi 7\Lib".
2. Start Delphi 7 IDE.
3. Open "AppBarD7.dpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 1
1. Unzip files from "BCB1" directory to your "CBuilder\Lib" directory.
2. Start C++ Builder IDE.
3. Select "Component \ Install..." menu item.
4. Press "Add" button and select "AppBar.dcu" file.
5. Rebuild library.

to C++ Builder 3
1. Unzip files from "BCB3" directory and copy them to "CBuilder3\Lib".
2. Start C++ Builder 3 IDE.
3. Open "AppBarCB3.bpk" file.
6. Select "Project \ Make AnimationEffectCB3" menu item.
7. Select "Component \ InstallPackages" menu item.
8. Press "Add" button and select "AnimationEffectCB3.bpl" file.

to C++ Builder 4
1. Unzip files from "BCB4" directory and copy them to "CBuilder4\Lib".
2. Start C++ Builder 4 IDE.
3. Open "AppBarCB4.bpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 5
1. Unzip files from "BCB5" directory and copy them to "CBuilder5\Lib".
2. Start C++ Builder 5 IDE.
3. Open "AppBarCB5.bpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 6

1. Unzip files from "BCB6" directory and copy them to "CBuilder6\Lib".
2. Start C++ Builder 6 IDE.
3. Open "AppBarCB6.bpk" file.
4. Install package to the components palette ("Install" button).

Source Code

1. Uninstall / delete all previous (trial) instances of AppBar component.
2. Unzip files from "Sources" directory and copy them to "..\Lib" directory.
3. Run Delphi or ++ Builder IDE.
4. Select "Component \ Install..." menu item.
5. Press "Add" button and select "_AppBarReg.pas" file.
6. Rebuild library.

3 Registration Information

AppBar component is SHAREWARE. This means that you can try it out for free, but if you like it and want to use it you have to register it with the author. Before continue read and accept license agreement please.

The only difference between the unregistered and registered versions is that the registered one has not message box with remind to register and works without Delphi (C++ Builder) running. You can also purchase the source code, if you would like to have it, and be able to compile or modify the AppBar on any 32bit version of Delphi or C++ Builder.

If you would like to use the AppBar and receive full, unrestricted version, priority support or even source code — you have to purchase proper license.

All prices in US dollars. Registering entitles you to unlimited support via E-Mail, minor version updates indefinitely and major version updates for 6 month from date of purchase.

Registration types:

Full, unrestricted version without source code:

Single user license:

- **https://secure.element5.com/register.html?productid=140733 - $34,95**

Site license:

- https://secure.element5.com/register.html?productid=140734 **- $129,95**

Full version including 100% Source Code:

Single user license:

- **https://secure.element5.com/register.html?productid=140735 - $89,95**

Site license:

- https://secure.element5.com/register.html?productid=140736 **- $259,95**

Comments

1. ***Site license*** covers a single organisation in one location (building complex). If you buy a site license, you may use the software in unlimited number of your company's computers withing this area. Site license is very cost-effective if you have many computers (many software developers).

See license agreement for more details.

4 License Agreement

Copyright

The AppBar component (software) is Copyright © 2000-2002, by Utilmind Solutions® (Utilmind). All rights reserved.

The authors - Utilmind Solutions® and Aleksey Kuznetsov (founder of Utilmind), exclusively own all copyrights to the Advanced Application Controls (AppControls) and all other products distributed by Utilmind Solutions®.

Liability disclaimer

THIS SOFTWARE IS DISTRIBUTED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Restrictions

You may not attempt to reverse compile, modify, translate or disassemble the software in whole or in part. You may not remove or modify any copyright notice or the method by which it may be invoked.

Operating license

Unregistered version

You may distribute the unregistered version of software freely, provided that all files are included and remain unmodified and that no extra files have been added to the package. You may not ask any money for the distribution. You may use the unregistered version of software free of charge for testing purposes, but if you want to use it for other purposes than testing - you have to register it with the author.

Registered version (single user license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use registered version of the software only by a single person, on a single computer at a time. You may physically transfer the software from one computer to another, provided that the software is used only by a single person, on a single computer at a time. In group projects where multiple persons will use the software, you must purchase an individual license for each member of the group or purchase site license. Use over a "local area network" (within the same locale) is permitted provided that the software is used only by a single person, on a single computer at a time. Use over a "wide area network" (outside the same locale) is strictly prohibited under any and all circumstances.

Registered version (site/team license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team only in one location (building complex). If you purchase a site license, you may use the program in an unlimited number of your company's computers within this area.

Registered version (Educational site license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend.

You have the non-exclusive right to use and transfer registered version of software on any number of computers by your educational organisation (school/college/university etc) in one location (building complex). If you buy a educational site license, you may use the program in an unlimited number of your edicational organisation's computers within this area.

Registered version (World-wide license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team world-wide. If your company has many branches even with thouthands of computers, world wide license covers them all.

Notes (clarification)
"Single-user license" means "single-developer license". "Site license" means that it can be used by any number of software developers within your company.
You can use purchased components in ANY number of your projects and deploy the "end-user" applications to ANY number of your users/customers without any additional royalty fees. However you are not permitted to distribute the component itself (the source code or .dcu files of components).

Back-up and transfer
You may make one copy of the software solely for "back-up" purposes, as prescribed by international copyright laws. You must reproduce and include the copyright notice on the back-up copy.

Terms
This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to author.

Other rights and restrictions
All other rights and restrictions not specifically granted in this license are reserved by authors.

5 Frequently Asked Questions

Answers to some questions which we receiving rather often

Q. When user doubleclicks the form, the AppBar will restore previous placement [*LastDockingPlace property*]. Is there a way to avoid re-positioning of the AppBar ?
A. To disallow restoring of previous placement on doubleclick, just hook OnDblClick event.

Q. I have docked the AppBar to the left border of screen on program startup but the form appears with the same width as in "floating" state!
A. Seems that your form contains a lot of non-resizeable (not aligned) controls and should have enough space to show them, however can not auto-scroll for displaying. Just set "AutoScroll" property of your form to True.

Q. I have docked the form to the screen edge, but seems I can not resize it. Your demo programs allows to resize docked appbars. I have also checked Enabled property of Sizing structure and it's True. What I doing wrong?
A. Sure, you can not resize forms with fixed placement (i.e.: BorderStyle = bsDialog or bsSingle). Just make your form resizeable, set BorderStyle property to bsSizeable or bsSizeToolWin.

Q. Is there a way to delay the popup when anchored to the edge of the screen [*when the AppBar is "autohidden"*]? (I've had a couple users complain that they moved their mouse to fast when trying to get to a desktop icon or somewhere else and went to far and caused the appbar to pop out even though they quickly moved it away)
A. Yes. You should hook OnUnhiding event and unhide the AppBar with delay (delay can be implemented with TTimer component), using ShowHiddenAppBar method. Please check out an **example** for more details.

Q. I have a form with AppBar on it. There is a panel on the form. This hides part of the form. When I try to drag the form to dock it to the side of the screen or to undock it obviously now it doesn't work. Could you suggest a way in which I could fire the event which docks, undocks and moves the form when the user clicks or tries to drag the panel component.
A. In **v1.7** we have added bonus component, **TDragPanel**. You can drop this panel to the AppBar and user will be able to move and dock the appbar dragging this panel.

6 Properties

6.1 AllowedEdges

Applies to
AppBar component.

Declaration

```
type
  TAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);
  TAppBarEdges = set of TAppBarPlacement;

property AllowedEdges: TAppBarEdges;
```

Description
The AllowedEdges property is the set of possible placements for the Application Desktop Toolbars and defines the screen edges to which the AppBar can be docked.

To allow the AppBar to be docked to the left and right edge of screen — make both `bpLeft` and `bpRight` True. To allow the AppBar to be docked to the top and bottom edges — make `bpTop` and `bpBottom` properties True. To disallow the "undocking" — set the `bpFloat` to False.

Example
For example, you would like to allow your AppBar to be docked to the top and bottom edges of screen, disallow docking to the left and right edges and disallow the floating. In other words, the form should be always docked to the top or bottom edge of screen only and can not be undocked.

In this case you should set `bpTop` and `bpBottom` to True and other settings to False:

```
acAppBar1.AllowedEdges  := [bpTop, bpBottom];
```

See also
Placement and LastDockingPlace properties.

6.2 AlwaysOnTop

Applies to
AppBar component.

Declaration
```
property AlwaysOnTop: Boolean;
```

Description
The AlwaysOnTop property ensures that the appbar is always visible, even when you run another programs in a maximized window. AlwaysOnTop determines whether the appbar can always stay on top over other windows. If AlwaysOnTop is True, system will place your form at the top of the Z order and your form will always on top over other windows. If AlwaysOnTop is False, another windows can be placed over your form.

See also
AutoHide property.

6.3 AutoHide

Applies to
AppBar component.

Declaration
```
property AutoHide: Boolean;
```

Description
The AutoHide property controls whether the AppBar can be hidden (reduced to edge of screen) when your program deactivates and mouse pointer moves outside of the AppBar form. The "autohidden" AppBar can be redisplayed when mouse point to the thin line at the screen edge.

When you set the AutoHide to True, make sure that the AlwaysOnTop are also True. If AutoHide is True but AlwaysOnTop is False, the AppBar will pop out below the others windows and will "sinks" behind them.

Note
If you wll try to dock the AppBar to the edge of screen with other, already anchored "autohidden" AppBar, component will be unable to hide. In this case AutoHide property value became False and the OnUnableToAutoHide event occurs.

See also
AlwaysOnTop and DontSlideOnTaskBarWhenAutoHide properties;
OnUnableToAutoHide, OnHiding and OnUnhiding events.

6.4 Docking

Applies to
AppBar component.

Declaration

```
type
  TAppBarDocking = class
  published
    property Enabled: Boolean;

    // all values in screen pixels
    property Height: Word;
    property Width: Word;
    property MaxHeight : Word;
    property MaxWidth: Word;
    property MinHeight : Word;
    property MinWidth: Word;
  end;
```

Description

The docking structure used to specify some rules for docked AppBars (when they are anchored to the screen edge).

Width, MaxWidth and MinWidth properties controls the width of the AppBar when it docked to left or right edge of screen.

Height, MaxHeight and MinHeight properties controls the height of the AppBar when it docked to top or bottom edge of screen.

The width and height of docked appbar can be specified and determinated using Width and Height properties. Width property controls current width of the appbar when it docked to the left or right edge of screen. Height property controls current current height when the appbar docked to top or bottom edge.

To enable or disable sizing restrictions — use Enabled property.

See also

FloatRestrictions structure, which specifies rules for undocked, "floating" forms.

6.4.1 Enabled

Applies to

all sub-structures of AppBar component.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property of Docking structure controls whether the AppBar should use docking rules specified by MaxWidth, MaxHeight, MinWidth and MinHeight properties.

6.4.2 Width

Applies to

AppBar component as subproperty of Docking structure.

Declaration

```
property Width: Word;
```

Description

The Width property determines current height of the AppBar when it docked to left or right edge of

screen.

See also
Height property of Docking structure.

6.4.3 Height

Applies to
AppBar component as subproperty of Docking structure.

Declaration
```
property Height: Word;
```

Description
The Height property determines current height of the AppBar when it docked to top or bottom edge of screen.

See also
Width property of Docking structure.

6.4.4 MaxWidth

Applies to
AppBar component as subproperty of Docking and FloatRestriction structures.

Declaration
```
property MaxWidth: Word;
```

Description
The MaxWidth property of Docking structure specifies maximum width for the appbars docked to left or right edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

6.4.5 MaxHeight

Applies to
AppBar component as subproperty of Docking and FloatRestriction structures.

Declaration
```
property MaxHeight: Word;
```

Description
The MaxHeight property of Docking structure specifies maximum height for the appbars docked to top or bottom edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

6.4.6 MinWidth

Applies to
AppBar component as subproperty of Docking and FloatRestriction structures.

Declaration
```
property MinWidth: Word;
```

Description
The MinWidth property of Docking structure specifies minimum width for the appbars docked to left or right edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

6.4.7 MinHeight

Applies to
AppBar component as subproperty of Docking and FloatRestriction structures.

Declaration
```
property MinHeight: Word;
```

Description
The MinHeight property of Docking structure specifies minimum height for the appbars docked to top or bottom edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

6.5 DontSlideOnTaskBarWhenAutoHide

Applies to
acAppBar component.

Declaration
```
property DontSlideOnTaskBarWhenAutoHide: Boolean;
```

Description
The DontSlideOnTaskBarWhenAutoHide property controls whether the AppBar should not appear over the taskbar when it slides from the screen side (when user activates it from hidden state).

By default, this property set to False, so it will slide from whole side of screen. Set it to False if you don't want to obscure other application desktop toolbars, when your AppBar became activated.

See also
AutoHide property.

6.6 FloatRestrictions

Applies to
AppBar component.

Declaration
```
type
```

```
TAppBarFloatRestrictions  = class
  published
    property Enabled: Boolean;

    property MaxWidth: Word;
    property MaxHeight : Word;
    property MinWidth: Word;
    property MinHeight : Word;
  end;
```

Description

The FloatRestrictions structure used to specify some restrictions to the form size, when the appbar in normal, "floating" state. To specify maximum and minimum form size use MaxWidth, MaxHeight, MinWidth and MinHeight properties.

To enable or disable floating restrictions — use Enabled property.

See also

Docking structure which specifies sizing rules for docked appbars.

6.6.1 Enabled

Applies to

all sub-structures of AppBar component.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property FolatRestrictions structure specifies whether you would like to use sizing restrictions for undocked, "floating" appbars. Set Enabled to True if you would like to specify maximum / minimum size for "floating" appbar.

6.6.2 MaxWidth

Applies to

AppBar component as subproperty of FloatRestictions and Docking structures.

Declaration

```
property MaxWidth: Word;
```

Description

The MaxWidth property controls the maximum width of your form. If size restrictions is Enabled and MaxWidth value other than 0, you will unable to make horizontal resize of your window with width value greater than specified in MaxWidth property.


If you would not to specify restriction for maximum form width, leave this value 0.

See also

MaxHeight property of FloatRestrictions structure.

6.6.3 MaxHeight

Applies to

AppBar component as subproperty of FloatRestictions and Docking structures.

Declaration

```
property MaxHeight: Word;
```

Description

The MaxHeight property controls the maximum height of your form. If size restrictions is Enabled and MaxHeight value other than 0, you will unable to make vertical resize of your window with height value greater than specified in MaxHeight property.

If you would not to specify restriction for maximum form height, leave this value 0.

See also

MaxWidth property of FloatRestrictions structure.

6.6.4 MinWidth

Applies to

AppBar component as subproperty of FloatRestictions and Docking structures.

Declaration

```
property MinWidth: Word;
```

Description

The MinWidth property controls the minimum width of your form. If size restrictions is Enabled and MinWidth value other than 0, you will unable to make horizontal resize of your window with width value lower than specified in MinWidth property.

If you would not to specify restriction for minimum form width, set this value to 0.

See also

MinHeight property of FloatRestrictions structure.

6.6.5 MinHeight

Applies to

AppBar component as subproperty of FloatRestictions and Docking structures.

Declaration

```
property MinHeight: Word;
```

Description

The MinHeight property controls the minimum height of your form. If size restrictions is Enabled and MinHeight value other than 0, you will unable to make vertical resize of your window with height value lower than specified in MinHeight property.

If you would not to specify restriction for minimum form height, set this value to 0.

See also

MinWidth property of FloatRestrictions structure.

6.7 LastDockingPlace

Applies to

AppBar component.

Declaration

```
type
  TAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);

property LastDockingPlace: TAppBarPlacement;
```

Description

The LastDockingPlace property determines previous placement of the AppBar. Previous AppBar placement always saved for further restoring it on doubleclick.

To prevent restoring previous placement on doubleclicking — hook OnDblClick event.

See also

Placement property and OnDblClick event.

6.8 Placement

Applies to

AppBar component.

Declaration

```
type
  TAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);

property Placement: TAppBarPlacement;
```

Description

The Placement property controls current placement of the AppBar on screen. When Placement is `bpLeft`, `bpRight`, `bpTop` or `bpBottom`, the AppBar is anchored to according screen edge. `Placement = bpFloat` means that form in currently undoked, normal "floating state".

When the AppBars in floating state, they are looks and feels like usual forms. They may have title bar, system menu and caption buttons. However, when the AppBars became docked, it lose all controls in non-client area and shifts all other application windows from screen edge giving enough space for the Application Toolbar (unless "autohideable" appbars with AutoHide = True).

User can change placement of the AppBar moving the form on screen with mouse, or you can modify it programatically at run-time, changing value of the Placement property.

After changing the Placement, previous property value will be assigned to LastDockingPlace property. Previous placement always saved for further restoring it on doubleclick.

See also

AutoHide, AlwaysOnTop and LastDockingPlace properties.
OnDblClick event.

6.9 RegistrySaver

Applies to

AppBar component.

Declaration

```
type
  TAppBarRegistrySaver  = class
  public
    procedure Load;
    procedure Save;
  published
    property Enabled: Boolean;
    property Storage: TAppBarStorageType ;

    property RegLocation : TAppBarRegLocation ;
    property RegKey: String;

    property IniFileName : String;
    property IniSection : String

    property RestoreProperties : TAppBarRestoreProperties ;
    property Options: TAppBarRegOptions;
  end;
```

Description

The AppBar component able to automatically save all settings (placement, autohide mode, topmost mode, docking/sizing rules etc) on closing the AppBar and restore them again on next program startup. For this purpose used RegistrySaver structure which, disite the name, can store AppBar settings either in the system registry and .INI file.

How to use ?

To make settings restored on every program startup — set Enabled to True.

To specify where to store the settings (in the system registry or in INI file) — use Storage property.

- If **`Storage = useRegistry`**, all settings will be stored in the system registry. In this case you must point exact location where to store settings in the RegLocation and RegKey properties. You don't need to specify IniFileName and IniSection values. Leave them blank.

- If **`Storage = useIniFile`**, settings will be stored in the INI file, specified in the IniFileName property. Section of the Ini-file where to store AppBar settings should be specified in the IniSection property. In this case you don't need to specify registry keys and you may leave RegKey and RegLocation properties blank.

To specify which settings should be restored on every startup — use RestoreProperties property.

By default the RegistrySaver update stored properties when the AppBar is about to be closed. If you would like to update saved settings after *every* movement or resizing — make `saveOnMoveResize` = True in the Options property.

6.9.1 Enabled

Applies to
all sub-structures of AppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of RegistrySaver structure controls whether you would like to restore AppBar settings on every program startup.

See also
RestoreProperties property.

6.9.2 IniFileName

Applies to
AppBar component as subproperty of RegistrySaver structure.

Declaration
```
property IniFileName: String;
```

Description
IniFileName property contains the name of the .INI file, where you would like to store all appbar settings. To specify contstant which identifies the section in the .INI file — use IniSection property.

Note
This property will be used only if you use .INI file as storage (`Storage = useIniFile`).

See also
IniSection and Storage properties of RegistrySaver structure.

6.9.3 IniSection

Applies to
AppBar component as subproperty of RegistrySaver structure.

Declaration
```
property IniSection: String;
```

Description
IniSection property specifies the constant which identifies the section in the .INI file, specified in the IniFileName property.

Note
This property will be used only if you use .INI file as storage (`Storage = useIniFile`).

See also
IniFileName and Storage properties of RegistrySaver structure.

6.9.4 Options

Applies to
AppBar component as subproperty of RegistrySaver structure.

Declaration
```
type
  TAppBarRegOptions = set of (saveOnClose,
                              saveOnMoveResize);

property Options: TAppBarRegOptions;
```

Description
The Options property controls when RegistrySaver should save the AppBar properties, enumerated in the RestoreProperties property. By default, AppBar saves configuration when your application is about to be closed (when `saveOnClose` is True). However, you can save all settings after every form movement or resizing, just set `saveOnMoveResize` to True.

See also
RestoreProperties property.

6.9.5 RegKey

Applies to
AppBar component as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the AppBar settings (placement / restrictions etc). All stored settings will be updated every time when user move or close the form (dependently of Options)

Default property value is `'\Software\CompanyName\ProgramName\AppBar'` . You should point the RegKey to location where stored the configuration settings of your own application. In example - `'\Software\UtilMind Solutions\The Cool Program\AppBar Settings'`.

Note
This property will be used only if you use system registry as storage (`Storage = useRegistry`).

See also
RegLocation property.

6.9.6 RegLocation

Applies to
AppBar component as subproperty of RegistrySaver structure.

Declaration
```
property RegLocation: TAppBarRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the

AppBar settings. RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When RegLocation is `rlCurrentUser`, acAppBar stores all settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

Note


This property will be used only if you use system registry as storage (`Storage = useRegistry`).

See also

RegKey property.

6.9.7 RestoreProperties

Applies to

AppBar component as subproperty of RegistrySaver structure.

Declaration

```
type
  TAppBarRestoreProperties = set of (rfAllowedEdges, rfFormSize,
rfDocking, rfFloatRestrictions, rfSizing, rfSliding);

property RestoreProperties: TAppBarRestoreProperties ;
```

Description

The RestoreProperties property of the RegistrySaver structure specifies which properties, saved after previous program execution, you would like to restore on every program startup.

Values	Meaning
`rfAllowedEdges`	restores value of AllowedEdges property when True;
`rfFormSize`	restores the form Placement and size when True;
`rfDocking`	restores Docking rules when True;
`rfFloatRestrictions`	restures FloatRestrictions;
`rfSizing`	restores Sizing rules;
`rfSliding`	restures Sliding settings.

See also

Options property.

6.9.8 Storage

Applies to

AppBar component as subproperty of RegistrySaver structure.

Declaration

```
type
  TAppBarStorageType = (useRegistry, useIniFile);

property Storage: TAppBarStorageType ;
```

Description

The Storage property specifies where to store AppBar settings, in the system registry or in the INI file.

- If **Storage = useRegistry**, all settings will be stored in the system registry. In this case you must point exact location where to store settings in the RegLocation and RegKey properties. You don't need to specify IniFileName and IniSection values. Leave them blank.

- If **Storage = useIniFile**, settings will be stored in the INI file, specified in the IniFileName property. Section of the Ini-file where to store AppBar settings should be specified in the IniSection property. In this case you don't need to specify registry keys and you may leave RegKey and RegLocation properties blank.

6.10 Sizing

Applies to
AppBar component.

Declaration
```
type
  TAppBarSizing = class
  published
    property Enabled: Boolean;

    property HorizontalIncrement : Word; // in pixels
    property VerticalIncrement : Word;
  end;
```

Description
The Sizing structure used to specify some rules on resizing of the appbars. You can specify horizontal and vertical increments (when user resize the appbar, it can be resized by specified number of pixels per mouse movement). To allow or disallow resizing of the AppBar — use Enabled property.

See also
FloatRestrictions structure.

6.10.1 Enabled

Applies to
all sub-structures of AppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of Sizing structure controls whether user can resize the AppBar window. When Enabled is False, user can NOT resize appbars, even if appbar in "floating" state and form's BorderStyle is [bsSizeable].

See also
HorizontalIncrement and VerticalIncrement properties of Sizing structure.

6.10.2 HorizontalIncrement

Applies to
AppBar component as subproperty of Sizing property.

Declaration
```
property HorizontalIncrement: Word; // 1 pixel by default
```

Description

The HorizontalIncrement controls the horizontal sizing increment for the appbars (either docked and "floating"). Usually you resize forms forms with 1 pixel precision. HorizontalIncrement property lets users to resize forms by specified number of pixels per mouse movement.

See also

VerticalIncrement property of Sizing structure.

6.10.3 VerticalIncrement

Applies to

AppBar component as subproperty of Sizing property.

Declaration

```
property VerticalIncrement: Word; // 1 pixel by default
```

Description

The VerticalIncrement controls the vertical sizing increment for the appbars (either docked and "floating"). Usually you resize forms forms with 1 pixel precision. VertocalIncrement property lets users to resize forms by specified number of pixels per mouse movement.

See also

HorizontalIncrement property of Sizing structure.

6.11 Sliding

Applies to

AppBar component.

Declaration

```
type
  TAppBarSliding = class
  published
    property Enabled: Boolean;
    property SlideTime : Word; // in milliseconds
  end;
```

Description

The sliding effect used to animate all automatic movements of the AppBar. This effect can be displayed when you programatically changing the Placement property, or when the hidden AppBar (with AutoHide = True), popup to the screen.

To enable or disable sliding effect — use Enabled property. To specify duration of effect — use SlideTime property.

6.11.1 Enabled

Applies to
all sub-structures of AppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of Sliding structure determines is the sliding effect currently enabled.

When the Enabled is True, all automatic movements of the AppBar will be displayed with smooth sliding effect.

To specify duration of the sliding effect — use SlideTime property.

See also
SlideTime property of Sliding structure.

6.11.2 SlideTime

Applies to
AppBar component as subproperty of Sliding structure.

Declaration
```
property SlideTime: Word; // in milliseconds
```

Description
The SlideTime property specifies duration of sliding effect, in milliseconds, which displayed on all automatic movements of the AppBar (if sliding effect is enabled).

See also
Enabled property of Sliding structure.

6.12 TaskIcon

Applies to
AppBar component.

Description
```
type
  TAppBarShowBehavior = (sbShowAlways, sbHideAlways, sbOnFloat);

property TaskIcon: TAppBarShowBehavior ;
```

Declaration
The TaskIcon property controls whether you would like to show the program icon on the taskbar when the AppBar docked to screen edge.

Default property value is `sbOnFloat`, that means that task icon should appears only when the AppBar is in undocked, "floating" state.

Set TaskIcon to `sbShowAlways` to make the task icon ALWAYS visible on the taskbar, even if AppBar is docked to screen edge. When TaskIcon is `sbHideAlways` — task icon will always hidden.

See also
TitleBar property.

6.13 TitleBar

Applies to
AppBar component.

Description

```
type
  TAppBarShowBehavior = (sbShowAlways, sbHideAlways, sbOnFloat);

property TitleBar: TAppBarShowBehavior ;
```

Declaration
The TitleBar property controls whether you would like to show the form's title bar (window caption) when the AppBar docked to screen edge.

Default property value is `sbOnFloat`, that means that title bar should appears only when the AppBar is in undocked, "floating" state.

Set TitleBar to `sbShowAlways` to make the tiele bar ALWAYS visible, even if AppBar is docked to screen edge. When TaskIcon is `sbHideAlways`, the title bar will be always hidden (captionless form).

See also
TaskIcon property.

7 Methods

7.1 IsMouseOverAppBar

Applies to
AppBar component.

Declaration

```
function IsMouseOverAppBar: Boolean;
```

Description
The IsMouseOverAppBar function returns True if mouse is over the form (AppBar) and False if mouse is outside of AppBar.

See also
Example

7.2 ShowHiddenAppBar

Applies to
AppBar component.

Declaration
```
procedure ShowHiddenAppBar(Show: Boolean);
```

Description
The ShowHiddenAppBar method pops up the hidden AppBar (when AutoHide property is True and anchored to the screen edge).

When *Show* parameter is True, AppBar unhides itself from behind the screen, and hides it back if *Show* parameter is False.

Remarks
When you call the ShowHiddenAppBar method, the OnHiding and OnUnhiding events will NOT occurs!

See also
AutoHide property; OnHiding and OnUnhiding events.

7.3 UpdateBar

Applies to
AppBar component.

Declaration
```
procedure UpdateBar;
```

Description
The UpdateBar method updates the form placement and repaints the AppBar on screen.

See also
Placement property.

8 Events

8.1 OnActivate

Applies to
AppBar component.

Declaration
```
property OnActivate: TNotifyEvent;
```

Description
The OnActivate event for an appbar occurs when the appbar becomes active (i.e. when user clicks the client area with a mouse). Your appbar becomes active when it is initially run or when focus is shifted from another form or Windows application to your form (AppBar).

See also
OnDeactivate event.

8.2 OnClick

Applies to
AppBar component.

Declaration
```
property OnClick: TNotifyEvent;
```

Description
The OnClick event occurs when the user clicks the form with AppBar component. Typically, this is when the user presses and releases the primary mouse button with the mouse pointer over the form.

See also
OnDblClick event.

8.3 OnDblClick

Applies to
AppBar component.

Declaration
```
property OnDblClick: TNotifyEvent;
```

Description
The OnDblClick event occurs when user double-clicks the primary mouse button over the form (AppBar). Use the OnDblClick event to write code that responds to the double-click event.


Usually, when user doubleclicks the Application Toolbar, the AppBar should restore its previous placement (see LastDockingPlace property). Hooking of the OnDblClick event will prevent the AppBar from auto-restoring previous placement.

See also
OnClick event and LastDockingPlace property.

8.4 OnDeactivate

Applies to
AppBar component.

Declaration
```
property OnDeactivate: TNotifyEvent;
```

Description
The OnDeactivate event occurs when the user switches from your form (AppBar) to another form or Windows application. Use the OnDeactive event to do any special processing you want to occur before the AppBar is deactivated.

See also
OnActivate event.

8.5 OnDocked

Applies to
AppBar component.

Declaration
```
type
  TAppBarDockedEvent = procedure(Sender: TObject; NewPlacement:
TAppBarPlacement) of object;

property OnDocked: TAppBarDockedEvent;
```

Description
The OnDocked event occurs when the form has been docked to the screen edge or has restored to normal, "floating" state. *NewPlacement* parameter shows new placement of the AppBar.


To prevent docking to specific screen edge — use AllowedEdges property or OnDocking event.

See also
AllowedEdges property and OnDocking, OnDockMove events.

8.6 OnDocking

Applies to
AppBar component.

Declaration
```
type
  TAppBarDockingEvent = procedure(Sender: TObject; var NewPlacement:
TAppBarPlacement) of object;

property OnDocking: TAppBarDockingEvent;
```

Description
The OnDocking event occurs when the AppBar is about to be docked to the screen edge or be undocked (restored to its normal, "floating" state). Use *NewPlacement* parameter to specify another appbar placement for docking.

See also
AllowedEdges property and OnDocked, OnDockMove events.

8.7 OnDockMove

Applies to
AppBar component.

Declaration
```
property OnDockMove: TAppBarDockedEvent;
```

Description
TheOnDockMove event occurs when user moves the AppBar to the screen edge with mouse.

See also
OnDocked, OnDocking, OnExitSizeMove events.

8.8 OnExitSizeMove

Applies to
AppBar component.

Declaration
```
property OnExitSizeMove: TNotifyEvent;
```

Description
The OnExitSizeMove enent occurs after it has exited the moving or sizing mode.

See also
OnDocked, OnDocking, OnDockMove events.

8.9 OnFullScreenApp

Applies to
AppBar component.

Declaration
```
type
  TAppBarFullScreenAppEvent = procedure(Sender: TObject; Open: Boolean)
of object;

property OnFullScreenApp: TAppBarFullScreenAppEvent;
```

Description
The OnFullScreenApp notifies the AppBar whether a full screen application window is open. When the full screen applications is opens, Open parameter will be True, or False otherwise.

See also
OnOtherAppBarPosChanged and OnWIndowArrange events.

8.10 OnHiding

Applies to
AppBar component.

Declaration
```
type
  TAppBarHidingEvent = procedure(Sender: TObject; var AllowOperation:
Boolean) of object;

property OnHiding: TAppBarHidingEvent;
```

Description
The OnHiding event occurs when the docked appbar with AutoHide = True is about to hide from screen. This event occurs when the appbar losing focus (user activates another program) and appbar disappears from screen.

Set *AllowOperation* parameter in the event handler to False to discard the operation. See an example for more details.

Remarks
The OnHiding and OnUnhiding events does not occurs when you call ShowHiddenAppBar method

programatically.

See also
AutoHide property and OnUnhiding event.

8.11 OnOtherAppBarPosChanged

Applies to
AppBar component.

Declaration
```
property OnOtherAppBarPosChanged: TNotifyEvent;
```

Description
The OnOtherAppBarPosChanged event occurs when other Application Toolbar application (i.e. taskbar or MS-Office panel) changes its placement on screen.

See also
OnFullScreenApp and OnWindowArrange events.

8.12 OnUnableToAutoHide

Applies to
AppBar component.

Declaration
```
property OnUnableToAutoHide: TNotifyEvent;
```

Description
The OnUnableToAutoHide event occurs when user trying to dock the "autohideable" AppBar (with AutoHide = True) to the screen edge where another hidden Application toolbar already docked. Unforunately, Windows can not dock more than one hidden AppBar per screen edge.

When AppBar is unable to auto-hide, the AutoHide property becames False.

See also
AutoHide property.

8.13 OnUnhiding

Applies to
AppBar component.

Declaration
```
type
  TAppBarHidingEvent = procedure(Sender: TObject; var AllowOperation:
Boolean) of object;

property OnUnhiding: TAppBarHidingEvent;
```

Description
The OnUnhiding event occurs when the docked and hidden appbar (with AutoHide = True) is about to appear on screen. This event occurs when user moves the mouse pointer over thin line on screen edge and appbar became visible.

Set *AllowOperation* parameter in the event handler to False to discard the operation. See an example for more details.

Remarks
The OnHiding and OnUnhiding events does not occurs when you call ShowHiddenAppBar method programatically.

See also
AutoHide property and OnHiding event.

8.14 OnWindowArrange

Applies to
AppBar component.

Declaration
```
type
  TAppBarWindowArrangeEvent = procedure(Sender: TObject; Beginning:
Boolean) of object;

property OnWindowArrange: TAppBarWindowArrangeEvent;
```

Description
The OnWindowArrange event occurs when AppBar receives system message about arranging windows.

See also
OnFullScreenApp and OnOtherAppBarPosChanged events.

9 DragPanel component (bonus)

9.1 TDragPanel

Overview
The DragPanel component is the panel which lets user to move and dock the owner form dragging the panel that covers this form; or simply move the panel within its container. Works either with normal or MDI forms. The DragPanel is great addition to AppBar.

How to use?
Just drop the TDragPanel on your form and specify the DragObject. Then run the project and try to move the form dragging the panel.

Additional / improved properties

DragObject	specifies which object should be moved when user dragging the panel with mouse. User can drag the owner form by panel, or the panel itself within its container;
DragCursor	specifies the cursor image when user dragging the form by panel.

9.2 Properties

9.2.1 DragCursor

Applies to
DragPanel component.

Declaration

```
property DragCursor: TCursor;
```

Description

The DragCursor property specifies the image for mouse cursor when user dragging the form by panel.

See also

DragObject property.

9.2.2 DragObject

Applies to

DragPanel component.

Declaration

```
type
  TDragPanelDragObject = (doForm, doPanel, doNone);

property DragObject: TDragPanelDragObject;
```

Description

The DragObject property specifies which object should be moved when user dragging the panel with mouse. There are possible values for DragObject:

Value	Meaning
doForm	the owner *form* will be moved, when user dragging the panel with mouse;
doPanel	the *panel* will be moved itself within its container;
doNone	*nothing happends* . The panel behave like usual TPanel component.

See also

DragCursor property.