msResize 4.5

Order form

Please print this form and send it to: M. Schiffer, Mattschoe-Moll-Weg 26, D - 52064 Aachen, Germany

You may also e-mail appropriate information to: Schiffer@comports.com

If you're anxious to send creditcard-information by e-mail, please ask for our fax-number.

* = Neccessary, **O** = Please check appropriate

* Name: _____
Company: _____
* Address: _____
* ZIP, City: _____
* Country: _____
E-mail: _____
Phone: _____

I'd like to order a single developer license for the latest version of msResize.ocx for 20 US$ (or 20 EURO / 35 DM)

Mutiple developer licenses (starting at 10 licenses) with reduced pricing available on request. All prices may be subject to change for later versions.

Please send me the professional version of the control and my personal license-file

O as a binary attachment to my above e-mail address
O on a 3.5"-floppy disk (1.44 MB) to my above address (additional 5 US$ / 5 EURO / 10 DM)

O I additionally order the control's sourcecode for 50 US$ (or 50 EURO / 100 DM)
I agree to using sourcecode for nothing else but making sure the control works as designed and doesn't intentionally perform harmful effects.

I'd like to pay the registration fee

O **with my creditcard** no. _____, expires: ____/____
O American Express **O** EUROCARD / MASTERCARD **O** VISA

Cardholder's name (if different from licensee): _____

O cash
Please find the sum attached in my envelope.

O Überweisung/Scheck (Germany only):
Ich habe einen entsprechenden Betrag auf Ihr Konto überwiesen oder einen Verrechnungsscheck beigelegt.
Bankverbindung: M. Schiffer, Konto 20 024 105 bei Sparkasse Aachen (BLZ 390 500 00)

_____ _____
Date Signature (cardholder if creditcard-order)

msResize
Ordering a license

If you want to use this control for your projects, you need to purchase a license for the professional version of this control. You will then be sent the latest professional version of the control and a personal license file (the latter must not be distributed with your application, it is your personified proof of license needed in designtime only).

You can also purchase the sourcecode for the latest version of the control (only in addition to the control itself) for the one and only reason to make sure the control is not performing unwanted effects or behaving evil. By ordering sourcecode, you explicitely agree to not using the sourcecode in any other way (especially but not exclusively using it in your projects, copying it, modifying it, talking or writing about it).

You can order your license by e-mail, by fax or by normal mail. For these possibilities, the appropriate addresses can be found on the order form which you should use for your order. If you want to order by fax, you'll be e-mailed a fax-number on request to Schiffer@comports.com.

A single developer license (1 developer, any amount of applications) costs only 20 US$. Sourcecode additionally costs 50 US$. Prices for multiple developer licenses are available on request. Prices may be subject to change for later versions.

For delivery on a 3.5"-floppy diskette (1.44 MB), an additional service fee of 5 US$ is added.

You may pay the registration fee

- with your creditcard (American Express, EUROCARD or MASTERCARD, VISA)
- cash (by sending the sum via postal mail to the address given on the order form)
- your local currency is accepted when sending cash
- german users only: Sie können eine Überweisung / einen Verrechnungsscheck zur Zahlung nutzen.

For details and to order, please see the order form now.

msResize
Shareware

This version of msResize is Shareware.

Shareware means as much as "try before you buy" so you exactly know what you get for your money - it does not mean that you get something for free. For this reason, it is usual that Shareware is limited in some kind of way.

msResize is not limited in any way, neither functionally nor may it only be used for a couple of times or a certain amount of time, apart from the Info-window (often referred to as a "nag-screen") appearing before your form is displayed if you run your project as a compiled executable. It will not appear at designtime (unless you request it by doubleclicking the "Info"-property in the properties window).

You indeed can completely finish developing your application using msResize without registering at all. msResize won't bother you with nag-screens or whatever unless you run your project as a compiled executable. Once you're getting near to distributing your application, you of course do need to register. But be warned:

Only current versions of msResize (and appropriate sourcecode if applicable) are being sold and backwards-compatibility, although one of the points with high priority for future versions of msResize, cannot be guaranteed. Additionally, pricing may change for future versions (if needed, the latest pricing can be requested by sending an e-mail to Schiffer@comports.com). You therefore might want to register early enough (how about now?) to make sure you don't run into problems later.

If you want to use msResize in your projects, you must order a license for the latest professional version of the control before you can distribute your application.

Misc: Overriding properties

For each single control on your form, you can determine to not change any of the properties Top, Left, Width, Height and FontSize (which are, of course, responsible for the control's appearance) and the other options msResize offers you. You can simply override msResize's form-global settings by using the Tag property for the control in question.

If you want all controls on your form to keep one (or more) of these properties as fixed values you should use the msResize-properties RepositionTop, RepositionLeft, ResizeHeight, ResizeWidth, ResizeFonts, ResizePictures and IncludeContainers. These should generally be set according to the wished behaviour of the majority of the controls on your form.

To exclude the recalculation of a property for a single control use its Tag-property for one (or a combination of) the following expressions:

- `NoTop` Top-property for this control remains unchanged (redundant if RepositionTop is set to False)
- `NoLeft` Left-property for this control remains unchanged (redundant if RepositionLeft is set to False)
- `NoHeight` Height-property for this control remains unchanged (redundant if ResizeHeight isd set to False)
- `NoWidth` Width-property for this control remains unchanged (redundant if ResizeWidth is set to False)
- `NoFont` FontSize-property for this control remains unchanged (redundant if ResizeFonts is set to False)
- `ResizeFont` FontSize-property for this control will be changed (redundant if ResizeFonts is set to True)
- `NoPicture` Image in this PictureBox will not be adapted (redundant if ResizePictures is set to False)
- `ResizePicture` Image in this PictureBox will be adapted (redundant if ResizePictures is set to True)
- `IncludeContainer` Controls within this container will be adapted (redundant if IncludeContainers is set to True)
- `ExcludeContainer` Controls within this container will not be adapted (redundant if IncludeContainers is set to False)

These expressions are not case-sensitive (so "**NOTOP**"is the same as "**NoTop**", "**notop**", ...). If you want to use a combination of these expressions, there is no rule at all, use them like what you think to be most comfortable for you. In other words: "**NoTop**, **NoWidth**" produce the same effect as "**NoTopNoWidth**", "**notopnowidth**", "Whatever I want - at least I know I want **notop**-property and **nowidth**-property to be recalculated", and so on. The appropriate expressions just need to be found at any position within the Tag-string.

Examples:

```
' To not change a horizontal scrollbar's
' height when the parent form is resized:
HScroll1.Tag = "NoHeight"

' To not change a picturebox's size, fontsize and
```

```
' image when the parent form is resized:
Picture1.Tag = "NoHeight, NoWidth, NoFont, NoPicture"

' To not change a button's position and (font-)size
' when the parent form is resized:
Command1.Tag = "NoTop, NoLeft, NoHeight, NoWidth, NoFont"

' To exclude controls contained in Picture1 from the
' resizement (notice Picture1 itself will be resized,
' unless you additionally specify NoTop, NoLeft,...):
Picture1.Tag = "ExcludeContainer"
```

Event: ReadFormData

Fires before msResize begins capturing the form's current layout. This infomation is later used for the adaption of controls. ReadFormData either fires automatically or whenever the FormLoaded-method is called.

Private Sub Resize1_ReadFormData(Form As Object)

Parameter	Meaning
Form	Always msResize's parent form.

Event: AutoResize

Fires before msResize begins adapting controls to fit the parent form's new dimensions.

Private Sub Resize1_AutoResize(Cancel As Boolean)

Parameter	Meaning
Cancel	If set to true within the event's code, msResize will not adapt the controls

Event: AdaptImage

Fires before the image in a PictureBox control gets resized

Private Sub Resize1_AdaptPicture(PictureBox As Object, Cancel As Boolean)

Parameter	Meaning
PictureBox	PictureBox control whose image is to be adapted next
Cancel	If set to true within the event's code, msResize will skip this image

Event: AdaptControl

Fires for each single control before msResize is adapting it to the form's new size.

Private Sub Resize1_AdaptControl(Control As Object, NewLeft As Single, NewTop As Single, NewHeight As Single, NewWidth As Single, Cancel As Boolean)

Parameter	**Meaning**
Control	Control which is to be adapted next
NewLeft, NewTop, NewWidth, NewHeight	New position and size for the control to be adapted. Changing any of these parameters makes msResize use your values when adapting the control
Cancel	If set to true within the event's code, msResize will skip this control (that is: msResize will not change its appearance)
Remarks:	If code in this event is incorrect, you'll experience some irritating optical effects. **Don't panic -** this is inevitable: If KeepVisible is set to True (default and recommended setting), msResize disallows Windows to update the form while controls are being adapted and refreshes the form afterwards. In case of an error in this event's code, msResize cannot refresh the form because msResize will be stopped by Windows. Anyway, Windows itself does take over control - sad news is that it doesn't care to repaint the form. Minimize and maximize all crude-looking Windows of your IDE if neccessary to force Windows to do so. To avoid this situation, you may want to set the KeepVisible-property to True while developing.

Example:

```
Private Sub Resize1_AdaptControl(Control As Object, NewLeft As Single,
NewTop As Single, NewWidth As Single, NewHeight As Single, Cancel As
Boolean)
' To keep Command1's position the same (not its size):
' Note: In this case, using Command1's Tag-property with
' ----- "NoLeft,NoTop" would be faster and do the same

     If Control Is Command1 Then
          NewLeft = Command1.Left
          NewTop = Command1.Top
     End If

End Sub
```

Method: CenterForm

Used to center the parent form on the screen.

Example:

```
Sub Form_Resize()
 Resize1.CenterForm
End Sub
```

Method: FormResized

Used to "manually" reposition and resize the controls that have been on the form at the time of the last call to the FormLoaded-method. Properties of msResize and Tag-entries of the controls on the form (see Misc: Overriding properties) determine the recalculated values for the controls' new properties Top, Left, Width and Height.

When running msResize in automatic mode, this method is automatically called.

Example:

```
Sub Form_Resize()
 Resize1.FormResized
End Sub
```

See also: FormLoaded , AutoResize , AdaptContol

Method: FormLoaded

Used to manually initialize msResize: The current layout of the form will be used as a basis for all later calls of the FormResized-method or automatic resizement. When this method is called, it overwrites its old layout information, even if the Enabled-property of msResize is set to False.

You will usually (but not neccessarily) call this method after the form has been initialized or painted. If you move controls from code, add controls to the form at runtime or change PictureBox-controls' content you should also call this method afterwards, otherwise the new layout will not be adjusted in a later call to the FormResized-method.

When running msResize in automatic mode, this method is automatically called the first time the user starts resizing the form.

Example:

```
Sub Form_Load()
 Resize1.FormLoaded
End Sub
```

See also: FormResized

Property: Version

Description: The version number of the msResize control. Read only.

Usage: [*form*.]*Resize1*.**Version**

Remarks: This property of course is read-only. If you need to check the version of msResize, remember to convert this string to an integer value if you want to compare it with an integer number.

Data Type: String

Property: Top

Description: Determines the distance between the internal upper edge of an object and the upper edge of its container.

Usage: [*form*.]*Resize1*.**Top** [=*x*]

Remarks: As the control is invisible at runtime - don't care about this property. It only relates to the control's position during designtime.

Data Type: Single

See also: Left

Property: Tag

Description: Stores any extra data needed for your program.

Usage: [*form*.]*Resize1.***Tag** [=*Expression*]

Remarks: By default, the Tag property is set to an empty string ("").

Don't mix up msResize's Tag-property with the Tag-properties of the controls on your form. Setting msResize's Tag-property has no effect at all.

Data Type: String

Property: ResizeWidth

Description: Specifies whether the widths of all controls on the form should be recalculated according to the form's new dimensions after a resizement.

Usage: [*form*.]*Resize1*.**ResizeWidth** [=*True*|*False*]

Setting: The ResizeWidth property settings are:

True	(Default) Widths of the form's controls are recalculated
False	Widths of the form's controls are not recalculated - the controls' horizontal expansions remain the same as before the resizement.

Remarks: You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

See also: ResizeHeight

Property: ResizePictures

Description: Specifies whether images contained in PictureBox controls are resized.

Usage: [*form*.]*Resize1*.**ResizePictures** [=*True*|*False*]

Setting: The ResizePictures property settings are:
True Images are resized
False (Default) Images are not resized

Remarks: Notice that msResize internally keeps a copy of every picture in a PictureBox control if you set this property to true. These copies are neccessary to avoid the "pixelization"-effect that would otherwise appear. This however, dependant on the pictures' colordepth, size and number, may comsume a lot of memory.

If you change this property from False to True during runtime, you must explicitely call the FormLoaded-method afterwards.

You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

See also: RefreshPictures , AdaptImage

Property: ResizeHeight

Description: Specifies whether the heights of all controls on the a form should be recalculated according to the form's new dimensions after a resizement.

Usage: [*form*.]*Resize1*.**ResizeHeight** [=*True*|*False*]

Setting: The ResizeHeight property settings are:

True	(Default) Heights of the form's controls are recalculated
False	Width of the form's controls are not recalculated - the controls' vertical expansions remain the same as before the resizement.

Remarks: You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

See also: RepositionLeft, RepositionTop, ResizeWidth

Property: ResizeFonts

Description: Specifies whether the fontsizes for controls (if applying) are recalculated depending to the controls' new width.

Usage: [*form*.]*Resize1*.**ResizeFonts** [=*True*|*False*]

Setting: The ResizeFonts property settings are:

True	Fontsizes for controls are recalculated depending on the controls' new width
False	(Default) Fontsizes are not recalculated

Remarks: Take care about the fonttype you use for a control if you want it to be recalculated: TrueType-fonts are highly recommended, as non-TrueTypes do not allow a satisfying scaling for any size - chosing a non TrueType-font will in almost any case lokk pretty pixelized.

You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

Property: RepositionTop

Description: Specifies whether the top (vertical) positions of all controls on the form should be recalculated according to the container's new dimensions after a resizement.

Usage: [*form*.]*Resize1*.**RepositionTop** [=*True*|*False*]

Setting: The RepositionTop property settings are:

True	(Default) Top positions of the form's controls are recalculated
False	Top positions of the form's controls are not recalculated - the controls' top positions remain the same as before the resizement.

Remarks: You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

See also: RepositionLeft, ResizeHeight, ResizeWidth

Property: RepositionLeft

Description: Specifies whether the left positions of all controls on the form should be recalculated according to the form's new dimensions after a resizement.

Usage: [*form*.]*Resize1*.**RepositionLeft** [=*True*|*False*]

Setting: The RepositionLeft property settings are:

True	(Default) Left positions of the form's controls are recalculated
False	Left positions of the form's controls are not recalculated - the controls' left positions remain the same as before the resizement.

Remarks: You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

See also: RepositionTop, ResizeHeight, ResizeWidth

Property: RefreshPictures

Description: Specifies whether images contained in PictureBox-Controls are copied to the internal picture buffer prior to each resizement.

Usage: [*form*.]*Resize1*.**RefreshPictures** [=*True*|*False*]

Setting: The RefreshPictures property settings are:

- **True** Images are copied to the internal picture-buffer prior to each resizement
- **False** (Default) Images are not copied to the internal picture-buffer prior to each resizement

Remarks: Set this property to True if you want to change contents of some PictureBox-controls from within your application (like drawing methods or loading other pictures). msResize will then refresh its internal picture-buffer prior to drawing a resized version of the picture after the resizement.

Leave this property set to False if you don't intend to change contents of PictureBox-controls or if the ResizePictures property is set to False. Another way of accomplishing this property's task is to add "RefreshPicture" to the PictureBox's Tag property - the latter is recommended if you want only one or a few PictureBox-controls to be updated (increases speed and available memory).

Notice that updating the internal picture-buffer often is critical: msResize will use the images' current size for the next Resize-operation. For the second next resize operation, msResize will first refresh the picture buffer again before resizing is started. This results in pixelization-effects: Imagine a 200x200-pixel image being resized down to a 2x2-pixel size and then getting resized back to 200x200 pixels - you'll have a 200x200-pixel image with only four different "very large pixels" instead of getting back your original image. To keep your original image in its quality during the last call to FormLoaded, do not set this property to true.

Data Type: Boolean

See also: ResizePictures , AdaptImage , FormLoaded

Property: Name

Description: Specifies the name used in code to identify a form, control, or data access object. Not available at run time.

Remarks: The default name for new objects is the kind of object plus a unique integer. For msResize, the first new instance of the control is named "Resize1" by default.

Data Type: String

Property: MinWidth

Description: Specifies a minimum width for the form.

Usage: [*form*.]*Resize1*.**MinWidth** [=*x*]

Setting: The MinWidth property settings are:

0	(Default) No minumum width for the parent form
<> 0	Minimum width for the parent form

Data Type: Single

If you don't want to set a minimum width for your form and still don't want your form to look ugly for small widths you might want to implement the following code in your Form_Resize-event. This code will horizontally reposition and resize the controls on your form only if a minimum width is provided (in this case 2000), otherwise it won't:

```
Resize1.RepositionLeft = (Me.Width > 2000)    ' True if Width > 2000,
Resize1.ResizeWidth = (Me.Width > 2000)              ' False otherwise.
```

See also: MaxHeight, MaxWidth, MinHeight

Property: MinHeight

Description: Specifies a minimum height for the form.

Usage: [*form*.]*Resize1*.**MinHeight** [=*y*]

Setting: The MinHeight property settings are:

0	(Default) No minumum height for the parent form
<> 0	Minimum height for the parent form

Data Type: Single

If you don't want to set a minimum height for your form and still don't want your form to look ugly for small heights you might want to implement the following code in your Form_Resize-event. This code will vertically reposition and resize the controls on your form only if a minimum height is provided (in this case 2000), otherwise it won't:

```
Resize1.RepositionTop = (Me.Height > 2000)    ' True if Height > 2000,
Resize1.ResizeHeight = (Me.Height > 2000)     ' False otherwise.
```

See also: MaxHeight, MaxWidth, MinWidth

Property: MaxWidth

Description: Specifies a maximum width for the form.

Usage: [*form*.]*Resize1*.**MaxWidth** [=*x*]

Setting: The MaxWidth property settings are:

0	(Default) No maximum width for the parent form
<> 0	Maximum width for the parent form

Data Type: Single

See also: MaxHeight, MinHeight, MinWidth

Property: MaxHeight

Description: Specifies a maximum height for the form.

Usage: [*form*.]*Resize1*.**MaxHeight** [=*y*]

Setting: The MaxHeight property settings are:

0	(Default) No maximum height for the parent form
<> 0	Maximum height for the parent form

Data Type: Single

See also: MaxWidth, MinHeight, MinWidth

Property: Left

Description: Determines the distance between the internal left edge of an object and the left edge of its container.

Usage: [*form*.]*Resize1*.**Left** [=*x*]

Remarks: As the control is invisible at runtime - don't care about this property. It only relates to the control's position during designtime.

Data Type: Single

See also: Top

Property: KeepVisible

Description: This property determines whether the form should be locked while resizing controls on it. This property should usually be set to False to dramatically improve performance and produce a much better optical effect (see the difference and you know what's meant).

You may however wish to set this property to True if you should experience problems about controls not being redrawn correctly (if so, please notify the author and describe your problem) or to avoid the redraw-problem in designtime if errors in the AdaptControl-event's code arise.

Setting: The KeepVisible property settings are:

True	The form does not get locked while adapting controls
False	(Default) The form gets locked while adapting controls

Usage: [*form*.]*Resize1*.**KeepVisible** [=*True*|*False*]

Data Type: Boolean

See also: AdaptControl

Property: Index

Description: Specifies the number that uniquely identifies a control in a control array. Available only if the control is part of a control array; read-only at run time.

Usage: [*form*.]*Resize1*[**(** *integer* **)**]**.Index**

Setting: The Index property settings are:

No value	(Default) Not part of an array
0 to 32,767	Part of an array. Specify an integer greater than or eqal to 0 to assign a control to a control array.

Data Type: Integer

See also: Name

Property: IncludeContainers

Description: Specifies whether controls within containers (PictureBoxes, Frames, Tab controls,...) on the form should be resized and repositioned as well.

Usage: [*form*.]*Resize1*.**IncludeContainers** [=*True*|*False*]

Setting: The IncludeContainers property settings are:

True	(Default) All controls, including those in containers, are resized
False	Only controls on msResize's parent form are resized, not such that are within containers on the form

Remarks: You can use a control's tag property to handle exceptions to this property's setting.

Data Type: Boolean

Property: Enabled

Description: Specifies whether the control's functionality is enabled or not.

Usage: [*form*.]*Resize1*.**Enabled** [=*True*|*False*]

Setting: The Enabled property settings are:

True	(Default) The control's functionality is enabled
False	The control's functionality is disabled

Remarks: If you disable the control's functionality by setting the Enabled property to False, any call to FormLoaded or FormResized will not have an effect.

Data Type: Boolean

See also: Automatic, FormLoaded, FormResized

Property: Automatic

Description: Specifies whether the control automatically does its work or not.

Usage: [*form*.]*Resize1*.**Automatic** [=*True*|*False*]

Setting: The Automatic property settings are:

True	(Default) The control works in automatic mode
False	The control works in manual mode

Remarks: If you use msResize in manual mode, you'll need to use its methods FormLoaded and FormResized which provide the basic functionality of msResize. When working in automatic mode, these two methods are automatically triggered by msResize itself. In automatic mode, FormLoaded is triggered as soon as the user starts resizing a form and FormResized is triggered before the form's original resize-event fires.

Data Type: Boolean

See also: Enabled, FormLoaded, FormResized

msResize
Reference

Properties

- Automatic
- Enabled
- IncludeContainers
- Index
- KeepVisible
- Left
- MaxHeight
- MaxWidth
- MinHeight
- MinWidth
- Name
- RefreshPictures
- RepositionLeft
- RepositionTop
- ResizeFonts
- ResizeHeight
- ResizePictures
- ResizeWidth
- Tag
- Top
- Version

Methods

- FormLoaded
- FormResized
- CenterForm

Events

- AdaptControl
- AdaptImage
- AutoResize
- ReadFormData

Misc

- Overriding properties

msResize
Introduction

What is msResize?

msResize is a 32bit ActiveX-component (OCX) that can save you a whole lot of work when it comes to producing an application with user-resizable forms.

You can simply put an instance of the control onto a form (it is invisible at runtime) and add two function-calls (which are control methods), one for getting the current layout (that will be the default layout for any resizing) and one that resizes and repositions the controls on the form msResize has been placed on.

The control also allows you to exclude a recalculation of position, size or fontsize for every single control (see Reference: Misc).

Quick tour

Handling the control is easy: After adding it to your project just place an instance of the control on the form you want to automatically be adjusted to the user's resize-actions. A first instance of the control will be named "Resize1" which is what will be referred to in all further description.

If you have set the control's Automatic-property to True (default), you're finished. Anyway, there may be numerous reasons for you to disable the automatic mode. In that case, the FormLoaded and FormResized-methods are needed to control the functionality:

There are two important methods for this control that provide its basic functionality: First you'll need to tell the control what state of your form you want it to use as layout for a later resizing of the form:

```
Resize1.FormLoaded
```

Although this method's name has been chosen because it will usually be called in the "Form_Load"-event (recommended as the last command in this procedure), you can of course nevertheless call this method whenever you want the control to capture a new layout (for example if you repositioned a control on your form from code and you want msResize to use this new setting, you should call this method after the change).

Notice that if you add controls to your form at runtime (that is: from code) you will need to call this method again to update the layout-information msResize uses.

The second important method of the control is the one that actually does the repositioning and resizing of the controls on the form:

```
Resize1.FormResized
```

Although this method's name has been chosen because it will usually be called in the "Form_Resize"-event (recommended as the last command in this event unless there is some other code that is dependant on your controls' new layout), you can of course nevertheless call this method whenever you want the control to perform a recalculation and repositioning/resizing of your form's controls.

You will want to suppress the recalculation of size and/or position of controls in some cases (as an example you will probably not want a vertical scrollbar to become wider when the user changes the

form's size, it would look pretty ugly). That's why you can use any control's Tag-property to exclude recalculation of position or size for this control. Please refer to Misc: Overriding properties on how to accomplish this.

msResize
System requirements

You can use msResize with any programming language or other product (e.g. Microsoft Visual Basic, ActiveX-supporting WebBrowsers, Microsoft Access 97, Borland Delphi, ...) that supports usage of OCX-controls. However, the runtime-library for Microsoft Visual Basic 5 (SP2 or above) is required for the control to work.

This runtime-library (MSVBVM50.DLL) is quite large, it has therefore not been added to this archive (among others, it is included with Microsoft Visual Basic 5.0 and Microsoft Internet Explorer 4.0 and above). If you don't have a copy of the runtime-library yet, you can download it from many places on the internet (for example from **http://www.winsite.com** or **http://www.simtel.net/simtel.net/** which are both mirrored all over the world).

msResize
The Windows Registry

An OCX-control must always be registered with the Microsoft Windows Registry before you can use it.

A setup for this control has not been included because it would have enlarged the archive too much. You will need to register the control manually to view the product sample. You will probably not need to register it manually if you first load it in your development environment (IDE) because most IDEs automatically register a control when it is used in the IDE for the first time.

To register the control with the Windows Registry (this must also be done by the setup-program you use to distibute your application using msResize - installation-wizards usually do this for you) just run REGISTER.BAT or execute this commandline:

```
regsvr32.exe [path]\msresize.ocx
```

Where [path] is the directory where the file msResize.ocx is located (the system-subdirectory of your Windows-directory is recommended, in that case [path] can be left out as well as for the case that the location of msResize.ocx is in the path-environment or in the Windows-directory).

Example: If the file msResize.ocx is located in C:\MYCONTROLS\, the call would be

```
regsvr32.exe c:\mycontrols\msresize.ocx
```

If you experience any problems first try unregistering the control and then registering it again (as described above): To unregister the control run UNREG.BAT or use:

```
regsvr32.exe -u [path]\msresize.ocx
```

You can use "Start", "Run..." in Windows' Explorer to type in the above commands. You can also simply "drag" the file msResize.ocx in the Windows Explorer over the file regsvr32.exe to register the control (unregistering however does not work this way).

msResize
Contact, Disclaimer

msResize is a member of the

Visit the family on the internet:
http://www.comports.com/Schiffer

If you have any questions not covered in this helpfile, have suggestions for future versions or new controls, don't hesitate to contact the author (if you have a problem, please include information about your development environment, your operationg system, the version of this control and details about your hardware as much as a close description of your problem - do not send files unless requested). Please notice that registered customers are supported with priority, but any incoming mail will be answered.

Mathias Schiffer
Mattschoe-Moll-Weg 26
D - 52064 Aachen
Germany

E-Mail: Schiffer@comports.com

Disclaimer

msResize 4.5

msResize is a 32bit ActiveX-control (OCX) for usage with any programming language capable of using OCX-extensions (especially but not limited to Microsoft Visual Basic).

It has been designed to automatically resize controls on a form when the user resizes the window.

Help topics

msResize is a member of the "Invisible Toolz Family". Visit the family on the internet: http://www.comports.com/Schiffer