The Desaware ActiveX Gallimaufry

Version 2.0

for Visual Basic

by

Desaware, Inc.

Rev 2.0.2 (7/01)

Table of Contents

Introduction

What is a Gallimaufry?

The Merriam Webster dictionary defines 'gallimaufry' as a hodge-podge. We came across the word during a search for a substitute for 'medley'. Maybe the staff was short of sleep that day, who can say? The name tickled our collective funny bone and seemed to be the perfect fit for this collection of ActiveX controls.

A QuickStart Manual

Unlike our other manuals where we attempt to include detailed descriptions, sample code and anything else that might be helpful, this is a 'condensed' version. We have included more detailed information on each of the controls in the Help file. In addition upon installation an .rtf version of the manual will be placed in the Gallimaufry directory (gallim) which can be printed for your use.

Customer Support

We at Desaware have one very simple company policy - we do our best to treat our customers as we would like to be treated (after all, we are programmers too).

If you have purchased this software directly from Desaware and you feel that Gallimaufry is not for you or you are otherwise dissatisfied, please feel free to return it for a full refund (if you purchased it elsewhere you will need to contact your dealer for return or refund information - also, we reserve the right to limit this offer to 30 days from the invoice date). Your satisfaction is important to us, and we are well aware that this is a very unusual product and not appropriate for everyone.

For information on customer support and last minute changes, refer to the readme.wri file on the Gallimaufry CD. This file is compatible with write.exe (included with each copy of Windows).

There is a saying in the software world that no non-trivial program is completely bug free. The corollary to that saying is that no program with more than 10 lines in it is non-trivial. Gallimaufry is emphatically non-trivial....

Gallimaufry has undergone extensive testing to make it as bug free as possible. Nevertheless, it is possible that some have crept through. Please write or send us a fax if you find one, and include all of the steps required to reproduce the problem. Also, if there are any files needed to reproduce the error, send them to us on a diskette. Once we are able to duplicate a problem, we will provide you with a fix as quickly as possible.

Please address all correspondence to:

Desaware Inc.
1100 East Hamilton Avenue, Suite 4
Campbell, CA 95008

Phone (408) 377-4770, Fax (408) 371-3530

Internet: http://www.desaware.com, or support@desaware.com

Register! Register! Register!

We've found that the person who ends up using a software package is frequently not the person who pruchased it. Therefore we really need your registration card. This will allow us to send you information about upgrades.

But we can't send this information to you without knowing who you are!

And please send us your suggestions for features that you would like to see in future editions of this product!

Installation

Gallimaufry comes with a Windows installation program. **Refer to the readme file on the CD for the latest information regarding installation.**

- Place your Gallimaufry CD in your CD ROM drive.
- The setup program should automatically start if your computer's autorun mode is on. Otherwise, use the Start Button's Run... command in Windows to run d:\setup.exe or e:\setup.exe (depending on the drive letter assigned to your CD ROM drive). You can also use Windows Explorer or the File Manager to run this program.
- Select the desired setup option to start the installation.
- The setup program will prompt you for a destination directory for the Gallimaufry files. The default directory is Gallim under your root directory.
- Follow any further directions in the setup program. A summary of files installed will appear in the install.log file in your Gallimaufry directory.

Installation programs are tricky - and we have found that occasionally a system is configured in such a way that the installation program fails. Please refer to the readme file for the latest information on these situations, and for instructions for manual installation.

The directory containing the Gallimaufry sample files may contain files **readme.txt** or **readme.wri**. These files, if present, will contain recent information that could not be incorporated into the manual at time of printing. Use the Windows 'Notepad' program to view readme.txt, and the Windows 'Write' program to view readme.wri.

dwBanner

This section details step-by-step instructions on selecting and placing the **Banner** control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your Form

1. Verify that the **Banner** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project. (If you wish, you may also open an existing project, as long as it has a regular non-MDI form.)
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry Banner Control**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The **Banner** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

Once the control is on the form, there are a number of design time properties that can be customized. To make text scroll inside the control, the only property that need be set is the **ScrollText** property. However, the appearance and behavior of the control can be easily modified in a variety of ways using the other properties provided in the property pages. To access them, select the control, and in the **Property Browser** window, double-click on the "Custom" entry.

Property	**Description**
ScrollText	This is the actual text that will move across the control. It can be set to any string.

Font	This can be set this to any font installed on your system, and customize options such as bold, italic, etc.
AutoSizeFont	Set this property to True and the text will appear in the largest font size which will still fit inside the control's borders.
BorderStyle	Set this property to 1 (Fixed Single) if the **Banner is** to have a border. Otherwise, set it to 0 (None).
BackColor, ForeColor	Set these colors according to how the **Banner** control should appear.
BackStyle	If this property is set to 0 (Transparent), then the **Banner** will have a transparent background. Anything under the control will show through, and can be interacted with the user. If this property is set to 1 (Opaque), then the control's background will be the **BackColor**.

When the program is run, text will continually scroll across the Banner.

DwCmnDlg

This section details step-by-step instructions on selecting the dwCmnDlg component in your project (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Selecting the Component in your Project

1. Verify that the **Common Dialog** component is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project.
3. Select the **References…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry Common Dialog Component**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.

Using the Common Dialog for the First Time

Declare an instance of the appropriate common dialog object to be used. You can declare the object with or without the **WithEvents** key word. Declaring the common dialog object with the **WithEvents** key word allows the object to trigger events if the **RaiseCallbackEvent** property is set to True.

Dim WithEvents dwCmdDialog As dwFileOpenSave

1. Create a new instance of the object using the **Set** key word.

   Set dwCmdDialog = New dwFileOpenSave

2. Set the common dialog object's **DlgWindowOwner** property to the window handle of the window that is to be disabled when the dialog is displayed. If this property is not set, then the common dialog will be owned by the desktop and will behave like a modeless dialog.

   dwCmdDialog.DlgWindowOwner = MainForm.hWnd

3. Set any additional properties as desired. When assigning the object's Flags property, the glmcd??EnableHook constant must be included, otherwise certain properties may not work. Each object is initialized with this flag.

   dwCmdDialog.Flags = glmcdOFNEnableHook Or glmcdOFNNoChangeDir Or glmcdOFNHideReadOnly

4. Call the Show… method.

   lres = dwCmdDialog.ShowOpen

5. Set the common dialog object to Nothing when no longer using the component.

   Set dwCmdDialog = Nothing

Replacing the Common Dialog ActiveX Control

1. Declare an instance of the appropriate common dialog object to be used, on the form on which the Common Dialog ActiveX control exists. If the Common Dialog ActiveX control is used to display more than one common dialog, you would declare the appropriate objects from the component. You can declare the object with or without the **WithEvents** key word. Declaring the common dialog object with the **WithEvents** key word allows the object to trigger events if the **RaiseCallbackEvent** property is set to True.

   Dim WithEvents dwCmdDialog As dwFileOpenSave

2. Create a new instance of the object using the **Set** key word. You can do this in the **Form_Load** event.

   Set dwCmdDialog = New dwFileOpenSave

3. Set the common dialog object's **DlgWindowOwner** property to the window handle of the form or container on which the Common Dialog ActiveX control resides. If this property is not set, then the common dialog will be owned by the desktop and will behave like a modeless dialog.

   dwCmdDialog.DlgWindowOwner = MainForm.hWnd

4. Assign the object's properties to the same values used by the Common Dialog ActiveX control. Include any additional properties as desired. Be sure to include properties set in design mode for the Common Dialog ActiveX control. When assigning the object's **Flags** property, the **glmcd??EnableHook** constant must be included, otherwise certain properties may not work. Each object is initialized with this flag.

dwCmdDialog.Flags = glmcdOFNEnableHook Or glmcdOFNNoChangeDir Or glmcdOFNHideReadOnly

5. Comment out the call to the Show… (or Action) method of the Common Dialog ActiveX control and call the object's Show… method instead. Note that the Desaware Gallimaufry Common Dialog does not support the previous **Action** property.

   lres = dwCmdDialog.ShowOpen

6. Set the common dialog object to Nothing when no longer using the component. You can do this during the **Form_Unload** event.

   Set dwCmdDialog = Nothing

When testing is completed and the replacement is successful, remove the Common Dialog ActiveX control from your form or container, then remove the control reference from your project.

dwHexEdit

This section details step-by-step instructions on selecting and placing the **HexEdit** control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your Form

1. Verify that the **HexEdit** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project. (If you wish, you may also open an existing project, as long as it has a regular non-MDI form.)
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry HexEdit Ccontrol'**; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The HexEdit control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon to make the control take up all the available space on the form).

Step 2 - Customizing the Control at Design Time

Once the control is on the form, there are a number of design time properties that can be customized. The control, at design time, is filled with a test matrix of 255 bytes, for every possible byte value and ASCII character.

Property	**Description**
AddressDisplay	Set this property to False if you want the leftmost pane containing the addresses to disappear.
ASCIIDisplay	Set this property to False if you want the rightmost pane containing characters to disappear.

ReadOnly	Set this property to False if you wish the user to be able to edit the data inside the control at run-time.
ExtendedSelection	Set this property to True if you wish the user to be able to select more than one byte at a time inside the control.
BackColor, SelectedBackColor, SelectedForeColor	Set these colors according to how you wish the **HexEdit** control to appear. The sample matrix on the form will reflect the changes so that you are aware of how the control will appear at run-time.

Step 3 - Filling the Control with Data at Run-time

The bytes which make up the control at design time will vanish and will be replaced by a lone 0 at run-time. In order to make the control display something more than that, a “Memory Buffer” must be created and its contents must be passed to the **HexEdit** control. The memory buffer that is almost always used (and should be used!) is an array of **Bytes**. Place the following code in the **Form_Load** event of your form:

```
Sub Form_Load()
    Dim MyByteArray() As Byte
    Dim Index As Integer, sLen As Integer
    Dim TheText As String

    TheText =       "These characters will be converted to " + _
                            "its ASCII-Integer equivalents and " + _
                            "placed, one by one, in the byte buffer."

    sLen = Length(TheText)
    ReDim MyByteArray(1 to sLen)
    ' Fill MyByteArray with the ascii values of each
    ' character in turn from TheText
    For Index = 1 to sLen
    MyByteArray(Index) = Asc(Mid$(TheText, Index, 1))
    Next Index

    ' Transfer the byte array data into the HexEdit control
    HexEdit1.ByteBuffer = MyByteArray
End Sub
```

This code will fill a byte buffer with the ASCII equivalent of some characters from a string and will pass the byte buffer to the **HexEdit** control. The **HexEdit** control will then automatically begin displaying this data, and depending on the value of **ReadOnly** property, may allow the user to edit it.

dwPerList

This section details step-by-step instructions on selecting and placing the **Perspective List** control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your Form

1. Verify that the **Perspective List** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project. (If you wish, you may also open an existing project, as long as it has a regular non-MDI form.)
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry Perspective List**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The **Perspective List** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

Once the control is on the form, there are a number of design time properties that can be customized. To make text recede inside the control, the only property that need be set is the **Text** property. However, the appearance and behavior of the control can be easily modified in a variety of ways using the other properties provided in the property pages. To access them, select the control, and in the **Property Browser** window, double-click on the "Custom" entry.

Property	Description
Text	This is the actual text that will move across the control. It can be set to any string.
Font	You can set this to any font installed on your system, and customize options such as bold, italic, etc.

Alignment	Use this property to determine how the text is aligned within the control. You can set this property to 0 (Left), 1 (Right), 2 (Center), or 3 (Justified).
BackColor, ForeColor	Set these colors according to how the **Perspective List** control should appear.
BorderStyle	Set this property to 1 (Fixed Single) if the **Perspective List** is to have a border. Otherwise, set it to 0 (None).
BackStyle	If this property is set to 0 (Transparent), then the **Perspective List** will have a transparent background. Anything under the control will show through, and can be interacted with by the user. If this property is set to 1 (Opaque), then the control's background will be the **BackColor**.

When you run your program, text will recede across the Perspective List.

dwRotatePic

This section details step-by-step instructions on selecting and placing the **RotatePic** control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your Form

1. Verify that the **RotatePic** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project. (If you wish, you may also open an existing project, as long as it has a regular non-MDI form.)
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry RotatePic Control**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The **RotatePic** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

Once the control is on the form, there are a number of design time properties that can be customized. To rotate a picture, all you really need to do is set the **Picture** and **Angle** properties. However, the appearance and behavior of the control can be easily modified in a variety of ways using the other properties provided in the property pages. To access them, select the control, and in the **Property Browser** window, double-click on the "Custom" entry.

Property	Description
Picture	Double-clicking on this property will bring up a file dialog, which you can use to select any bitmap file. Once you have chosen a bitmap, it will appear in the **RotatePic** control.

PictureName	You can set this property to the path name or IP address of a bitmap file. This is useful for ActiveX containers such as Internet Explorer, which have difficulty with the standard Picture object.
Angle	Setting this property does the actual rotation. It can be any integer from 0 to 359. Once you have changed this property, the picture in the **RotatePic** control will be immediately rotated.
AutoSize	Set this property to True if you want the **RotatePic** control to make itself the same size as the picture it contains.
BorderStyle	Set this property to 1 (Fixed Single) if you want the **RotatePic** to have a border around it. Otherwise, set it to 0 (None).
BackColor	Set this to the color you want to use for the background of the **RotatePic** control.
BackStyle	If this property is set to 0 (Transparent), then the **RotatePic** will have a transparent background. Anything under the control will show through, and can be interacted with by the user. If this property is set to 1 (Opaque), then the control's background will be the **RotatePic**.

dwSpiralBox

This section details step-by-step instructions on selecting and placing the **SpiralBox** Control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your Form

1. Verify that the **SpiralBox** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project. (If you wish, you may also open an existing project, as long as it has a regular non-MDI form.)
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry SpiralBox Control**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The **SpiralBox** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

You can customize the appearance of the SpiralBox control using these properties:

Property	**Description**
BorderStyle	Set this property to 1 (Fixed Single) if you wish the SpiralBox to have a border. Otherwise, set it to 0 (None).
BackColor	This property is the color that will appear in the background of the SpiralBox control.

Step 3 - Programming the SpiralBox Control for Run-time Actions

Spirals are added to the **SpiralBox** control during run-time. This is done by using the **AddItem** method. For example,

```
{name of your SpiralBox control}.AddItem vbBlue, .8, .3
```

The first value you need to pass to **AddItem** is a color. This is the color that will be used to draw the new spiral. The second two values are decimal values from .001 to .999, which determine how the spiral is drawn. For more information on these values, refer to the Help File.

NOTE: If you are using any environment other than Visual Basic 5.0 or 6.0, then you will need to add a fourth value when using **AddItem**. This value is a positive integer, which determines the "order" of the new spiral. A value of 0 will place the new spiral at the bottom of the "pile" of spirals, and higher values can be used to place the new spiral "in between" others, or on top.

dwTaskbar

This section details step-by-step instructions on selecting and placing the **MDI Taskbar** control on your form (in Visual Basic) in addition to some of its basic functionality and properties/methods.

Step 1 - Placing the Control on your MDI Parent Form

1. Verify that the **MDI Taskbar** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** install program, you should be fine.
2. Start Visual Basic, and create a new project.
3. Select the **Components…** item from the **Project** menu.
4. Scroll down the list of controls to the '**Desaware Gallimaufry MDI Taskbar**'; select the checkbox to the left.
5. Click on the **OK** button to exit the dialog box.
6. The **MDI Taskbar** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on your MDI Parent form (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

To use the MDI Taskbar, all you really need to do is drop it onto the MDI Parent Form of any MDI project, and run the program. However, the appearance and behavior of the control can be easily modified in a variety of ways using the other properties provided in the property pages. To access them, select the control, and in the **Property Browser** window, double-click on the "Custom" entry.

Property	Description
BackColor	This is the color of the **MDI Taskbar**.
ButtonColor	This is the color of each button on the **MDI Taskbar**.

ButtonWidth	This is the width, in twips, of each button on the **MDI Taskbar**. The buttons will shrink during run-time if there is not enough room for all of them, but **ButtonWidth** is the default value.

dwTwain TWAIN Scanner Control - QuickStart

The Desaware TWAIN scanning Control (dwTwain) is an easy to use scanning control that is able to acquire images from most TWAIN compliant devices. The control supports the following features:

- ♦ Add scanning support to any Visual Basic application (or other applications that can act as ActiveX control containers).
- ♦ Supports TWAIN compliant devices including scanners, digital cameras and many video frame capture devices.
- ♦ Capture to memory or directly to disk (if supported by the image source).
- ♦ Auto capture of multiple images (if supported by the image source).
- ♦ Direct access to the TWAIN driver eliminates the need to install separate application software such as the Microsoft imaging system.
- ♦ Easy to use methods compatible with the TWAIN standard "Select Source" and "Acquire" commands.
- ♦ No run-time fees for distributing the component with your compiled application.

Step 1 - Placing the Control on Your Form

Make sure the **Desaware TWAIN Scanning** control is already installed; if you have already run the **Desaware ActiveX Gallimaufry** Install program, you should be fine.

You'll also need to have a TWAIN compliant scanner or imaging device installed. Virtually all scanners are TWAIN compliant and include the necessary drivers. Many digital cameras also come with TWAIN compliant drivers or have drivers available to download from the camera vendor's web site. Many video frame grabbers include TWAIN drivers as well. You can find sample TWAIN drivers for experimentation on the TWAIN web site at www.twain.org.

1. Start Visual Basic, and create a new project. (If you wish, you can also open an existing project, as long as it has a regular non-MDI form.)

2. Select the **Components…** item from the **Project** menu.
3. Scroll down the list of controls until you see '**Desaware TWAIN Scanning Control**'; select the checkbox to the left of this name.
4. Press the **OK** button to exit the dialog box.
5. The **TWAIN scanning** control icon should now appear at the bottom of your toolbox. You can click on the icon and draw the control on the form as normal (or double-click on the icon).

Step 2 - Customizing the Control at Design Time

Once the control is on the form, there are a number of design-time properties that can be customized. You do not need to set any of these properties to use the control. The most frequently used properties are those that allow you to specify your product's name and version. This information is used by the TWAIN driver to let the user know into which application they are scanning an image.

You can easily change these and other properties using the property pages provided. To access them, select the control, and in the Property Browser window, double-click on the "Custom" entry.

Manufacturer – Set this property to the name of your company.

ProductName – Set this property to the name of your product.

Step 3 - Adding Run-Time Code

Most TWAIN applications expose two commands to the user, one which allows them to select a device source (necessary because you may have more than one scanner installed on your system), the other that starts image acquisition.

To allow the user to select a source (assuming your scanning control is named TwainControl1, use the following code:

```
TwainControl1.OpenAndSelectScanner
```

To bring up the scanner's user interface and allow the user to acquire an image, use the following code:

```
TwainControl1.ScanImageWithUI
```

The ImageScanned event will be raised when an image is acquired.

When you are finished scanning, or which to close the TWAIN user interface, use the following code:

```
TwainControl1.CloseScanner
```

Refer to the TwainTest application for a more sophisticated sample program that includes error handling.

Other Sources of Information

Here are several other resources that we recommend for advance Windows development.

Visual Basic.NET or C# ... Which to Choose?

In this new eBook you will find an in-depth comparison of the two languages. In a feature-by-feature, head-to-head contest, Dan pulls no punches in calling the winner in each case.

But a technical comparison is only the beginning. With a keen eye for the business issues involved in language choice, the author focuses on the economic issues involved in this decision, considering the cost of retraining and long-term support, as well as that of initial development.

This ebook is available from mightywords.com.

Introduction to NT/2000 Security Programming with Visual Basic

NT Security is a subject that is intimidating, to say the least. But if you dig past the confusing acronyms, you'll find that it's actually very easy to understand. This article, based on Dan Appleman’s well received talks at VBits, will help you get started on the right foot with NT security, and give you the foundation of knowledge you'll need to understand even the most obscure security concepts. It will also introduce you to techniques for adding security based features to your applications (with an emphasis on Visual Basic applications).

This ebook is available from mightywords.com.

Moving to VB.net:Strategies, Concepts and Code

Written by Daniel Appleman (president of Desaware) and published by Apress (ISBN 1893115976).

VB.Net is not Visual Basic. Porting is stupid. COM is "dead". These are just a few of the things you'll learn as Dan takes you on a journey unlike any other into the world of VB.Net. Covers adoption strategies, unlearning VB6 concepts that are fatal in VB.Net, and analysis of language changes that goes beyond the documentation.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770 or email support@desaware.com.

Dan Appleman's Visual Basic Programmer's Guide To The Win32 API

Written by Daniel Appleman (president of Desaware) and published by McMillan, (ISBN 0-672-31590-4) - this sequel to the original 16 bit API Guide applies the same philosophy to teaching the Win32 API to developers using Visual Basic and VBA based applications. With more examples, more functions, more tutorial style explanations and a full text searchable electronic edition on CD-ROM, this book should prove a worthy successor to the 16 bit API book. Covers Visual Basic version 4 through 6.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770 or email support@desaware.com.

An upgrade CD is available for owners of the “PC Magazine's Visual Basic Programmer's Guide to the Win32 API” ISBN: 1-56276-287-7 for $24.99 + s&h directly from Desaware. Refer to our web site at www.desaware.com for additional information.

Dan Appleman's Developing COM/ActiveX Components with Visual Basic 6.0: A Guide to the Perplexed

Written by Daniel Appleman (president of Desaware) and published by McMillan, (ISBN 1-56276-576-0) - this book is designed for those programmers interested in using Visual Basic's object oriented technology to develop ActiveX components including EXE and DLL servers, ActiveX controls and ActiveX documents. Unlike many books that simply rehash the Visual Basic documentation, this one serves as a commentary to clarify and extend the documentation. Of special interest to VersionStamper customers will be the chapters on OLE and COM technology that will help them further understand the process of registering components, and the chapters on versioning and licensing.

The VB6 version also includes two new chapters on IIS Application development.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770 or email support@desaware.com.

Dan Appleman's Win32 API Puzzle Book and Tutorial for Visual Basic Programmers

Written by Daniel Appleman (president of Desaware) and published by Apress (ISBN# 1-893115-01-1). Appleman's Win32 API Guide covers 700 API functions. This book covers the other 7800. How? By teaching you everything you need to know to read and understand the Microsoft C documentation and create correct API declarations for use in Visual Basic. Presented in an entertaining puzzle/solution format that challenges you to solve real world API problems. In depth tutorials take you behind the scenes to understand what really happens when you call an API function from VB.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770 or email support@desaware.com.

How Computer Programming Works

A useful book for future programmers or anyone interested in explaining important computer programming concepts. Full color illustrations help to visually explain important topics! New expanded section on computer programming for the Internet.

Just as a child must learn the alphabet before they can read, future programmers must understand certain concepts before they can write their first program. This unique book uses full color illustrations to help the reader to truly understand the underlying computer science on which all programming is based. ISBN 1-893115-23-2.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770 or email support@desaware.com.

The Desaware Visual Basic Bulletin

and other related technical articles. At the Desaware website: http://www.desaware.com.

PC Magazine's Visual Basic Programmer's Guide To The Windows API

Written by Daniel Appleman (president of Desaware) this book is intended to help Visual Basic programmers navigate the complexities of Windows. It is the only text on Windows that is designed specifically for Visual Basic programmers, and the only one that covers the interactions between Visual Basic and Windows.

Available on CD Rom only from Desaware. Call (408) 377-4770 or email support@desaware.com.

Msdn.microsoft.com

This web site is a comprehensive reference.

Desaware Product Descriptions

Other features are the NT Service Toolkit *Light Edition*. This application is a subset of the Desaware NT Service Toolkit product. It allows a developer to create true NT services using Visual Basic. A background thread component that allows you to easily create objects that run in a separate background thread.

It also contains extensive sample code and three product updates.

1. The Professional Edition includes the Winsock Library, NT Service support and many other additional features & samples, plus three free updates. SpyWorks 2.1 (VBX Edition) is included in the Pro Edition.
2. SpyWorks Standard is a subset of Professional. A feature comparison is available on our web site.
3. Supports VB 4, 5 & 6, Windows 95, 98, 2000 & NT.

VERSIONSTAMPER 6.5

Distributing Component-Based Applications? Beware DLL HELL!

You've distributed your application and it's working fine. But your end user is still in charge of their system. What happens when they install a program that overwrites a component that your software needs to run? Can you verify that your users have the correct files required by your application? Can you really afford to spend two hours on the phone trying to figure out exactly what went wrong? Now you can easily avoid component incompatibilities by adding VersionStamper to your toolkit. It lets you check the versions of your program's components on your end user’s system, and correct the problem.

You are in control!

DLL Hell is a big problem, and with VersionStamper you can be in control of how this problem is detected and corrected. You determine dependency scanning (file size, date, version or other parameter), how and when the dependency scanning is done (upon start up, at midnight, at user's discretion), and how you want the problem resolved (automatically, an email message to your help desk, from a dependency list on your web site and more). This means you can handle versioning problems as simply as using a message box to call tech support, or even automatically updating the invalid components over the internet or corporate network. Imagine your application updating itself without user (or programmer) intervention! Imagine the hours and money saved in tech support calls! You can even use VersionStamper for incremental updates and bug fixes.

Is This For Real?

No, you don't have to pay a fortune in distribution fees - there are no run-time licensing fees. VersionStamper comes with a great deal of sample code. Don't distribute a component-based application without it!

4. Checks the versions of your dependent files and notifies you or the user of potential problems.
5. Internet extensions allow you to update versions across the Internet/intranets.
6. Cool and USEFUL sample programs show you how it works.

Includes VB source code for the VersionStamper components that you can use in your applications.

NT SERVICE TOOLKIT 1.1

Create a fully featured service in minutes using Visual Basic – even debug your service using the Visual Basic environment! Supports all NT service options and controls. Adheres to all Visual Basic threading rules. Background thread support allows easy waiting on system and synchronization objects. Client requests supported on independent threads for excellent scalability, with client impersonation available allowing services to act on behalf of clients in their own security context. Client requests and service control possible via COM/COM+/DCOM.

Simulation mode for testing as an independent executable. Create control panel applets for service control and other purposes.

DESAWARE EVENT LOG TOOLKIT 1.0

Visual Basic allows you to log events to the NT/2000 event log, but does not allow you to create custom event sources - so every event belongs to the application VB runtime, descriptions are limited, and event categories unavailable. Even if you use the API to log events, creating custom event sources for your application is not supported by VB, and is difficult with C++.

Desaware's new Event Log Toolkit makes creation of event sources easy, and provides all the tools needed to create and log custom events. Now your applications and services can support event logs in a professional manner, as recommended by Microsoft

STORAGETOOLS ver 2.5

StorageTools is your key to the OLE 2.0 Structured Storage Technology. Structured Storage allows you to create files that organize complex data easily in a hierarchical system. It is like having an entire file system in each file. OLE 2.0 takes care of allocating and freeing space within a file, so just as you need not concern yourself with the physical placement of files on disk, you can also disregard the actual location of data in the file. Additionally, with its support for transactioning you can easily implement undo operations and incremental saves in your application. StorageTools allows you to take advantage of the same file storage system used by Microsoft's own applications. In fact, we include programs (with Visual Basic source code) that let you examine the structure of any OLE 2.0 based file so that you can see exactly how they do it!

StorageTools includes documentation and controls to make it easy to work with the registration database under Windows 3.1, Windows NT & Windows 95/98 and 2000. For Visual Basic 4-6. We also include a simple resource compiler (with Visual Basic source code) so that you can create your own .RES files for use with Visual Basic.

StorageTools version 2.5 also includes the Desaware File Property component.

StorageTools includes 16 & 32 bit ActiveX/ATL controls, extensive documentation and sample code.

DESAWARE ACTIVEX GALLIMAUFRY Ver. 2

What is it?

gal·li·mau·fry (gàl´e-mô¹frê) noun
plural gal·li·mau·fries
A jumble; a hodgepodge.

[French galimafrée, from Old French galimafree, sauce, ragout : probably galer, to make merry. See GALLANT + mafrer, to gorge oneself (from Middle Dutch moffelen, to open one's mouth wide, of imitative origin).]
(From The American Heritage® Dictionary of the English Language, Third Edition copyright © 1992 by Houghton Mifflin Company)

What does a Twain control, spiral art program, set of linked list classes, a quick sort routine, a hex editor and a myriad of other custom controls have in
common?

They are all part of Desaware's ActiveX Gallimaufry.

You'll find most of these controls useful, the rest entertaining – but we guarantee that you'll find them all educational, because they come with complete Visual Basic 6.0 source code.

Curious?

Want to learn some advanced API programming techniques? Visit our web site for a full description and demo.

THE CUSTOM CONTROL FACTORY V 4.0
The Custom Control Factory is a powerful tool for creating your own animated buttons, multiple state buttons, toolbars and enhanced button style controls in Visual Basic and other OLE control clients, without programming. With 256 & 24 bit color support, automatic 3D backgrounds, image compression, over 50 sample controls and more. Plus MList2 - an enhanced listbox control. 16 & 32 bit ActiveX controls and 16 bit VBXs included.