The Desaware File Property Component™

Version 1.0

for Visual Basic

by

Desaware, Inc.

Rev: 1.0 (9/00)

Desaware, Inc.
Software License

Please read this agreement. If you do not agree to the terms of this license, promptly return the disk package and all accompanying items to the place from which you obtained them for a full refund.

This software is protected by United States copyright laws and international treaty provisions.

This program is licensed to you for your use. If you, personally, have more than one computer, you may install it on all your computers as long as there is no possibility of it being used at one location or on one computer while it is being used at another. Just as a book cannot be read by two different people in two different places at the same time, neither can this software be used by two different people in two different places at the same time. You may (and should) make archival copies of the software for backup purposes.

You may transfer this software and license as long as you include this license, manual, the software and all other materials and retain no copies, and the recipient agrees to the terms of this agreement.

You may not make copies of this software for other people or make copies of the manual. Companies or schools interested in multiple copy licenses or site licenses should contact Desaware directly at (408) 377-4770.

If you use this product to create a compiled Visual Basic program that you will distribute as an executable (.exe) file, you may include with your program a copy of the files dwProp.dll (the file property component). In this case you must include a valid copyright notice on all copies of your program. This can be either your own copyright notice, or the copyright notice that is on each Desaware File Property Component medium. You may not modify the file dwProp.dll in any way.

Limited Warranty

Desaware, Inc. warrants the physical medium and documentation enclosed herein to be free of defects in materials and workmanship for a period of sixty days from the purchase date.

The entire and exclusive liability and remedy for breach of this Limited Warranty shall be limited to replacement of defective CD(s) or documentation and shall not include or extend to any claim for or right to recover any other damages, including but not limited to, loss of profit, data or use of the software, or special, incidental or consequential damages or other similar claims, even if Desaware, Inc. has been specifically advised of the possibility of such damages. In no event will Desaware, Inc. 's liability for any damages to you or any other person ever exceed the suggested list price or actual price paid for the license to use the software, regardless of any form of the claim.

DESAWARE, INC. SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Specifically, Desaware, Inc. makes no representation or warranty that the software is fit for any particular purpose and any implied warranty of MERCHANTABILITY is limited to the sixty-day duration of the Limited Warranty covering the physical diskettes and documentation only (not the software) and is otherwise expressly and specifically disclaimed.

This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

This License and Limited Warranty shall be construed, interpreted and governed by the laws of the State of California, and any action hereunder shall be brought only in California. If any provision is found void, invalid or unenforceable it will not affect the validity of the balance of this License and Limited Warranty which shall remain valid and enforceable according to its terms.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/Manufacturer is Desaware, Inc., 1100 East Hamilton Ave. Suite 4, Campbell, CA 95008.

Table of Contents

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).

The Desaware File Property Component is a sophisticated product, and while we have checked it in as many environments as possible, it is conceivable that some bugs slipped through our testing process. Once we are able to duplicate a problem, we will provide you with a fix as quickly as possible.

If you have purchased this software directly from Desaware and you feel that File Property Component 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.

Year 2000 Compliance

We understand our customers' concern regarding the Year 2000 and the possible ramifications of the upcoming date change. While this product does meet the generally accepted definition of 'Year 2000 Compliant', due to certain limitations in Windows, the date for certain controls and/or functions must be between the years 1980 and 2107. Below is a listing of those functions affected by this limitation. In addition this information is noted in the manual.

dwProp.UserAdd

dwProp.UserSet

Please be advised that none of our other products (to date) are affected in any way. Year 2000 compliance information can also be found on our web site at www.desaware.comunder the heading YEAR 2000.

File Descriptions and Redistribution Terms

The following files and controls may be distributed with your compiled Visual Basic application without payment of license fees according to the terms on the License Agreement.

dwProp.dll	The file property component. It is usually installed in the System directory. It will be registered in the Registry by your installation program.
OLE32.dll	This file contains the core of OLE. The component versions of the StorageTools controls (dwStg.dll, dwReg.dll and dwProp.dll) require a version later than 4.00.

The following files may NOT be distributed and are subject to the terms in your license agreement. However, you may incorporate the source code from these files (where applicable) into your own programs.

PropDes.dll	Design time license file for the File Property Component.
PropSample.vbp and associated files.	Simple program that shows how to use the control.
PropTest.vbp, TestApp.vbp, UserControl.ctl and associated files.	Sample program that demonstrated using dwProp.dll in Visual Basic ActiveX controls.
Propbrowser.vbp and associated files.	Sample program that uses dwProp.dll to search directories for files with particular properties.
DwProp.hlp	On-line help file for the File Property Component.
DwLicGen.exe	Utility that creates license keys. Used when you use dwProp.dll as part of your own components.

Distribution of Applications That Use the File Property Component

When you ship programs that use our components, you have to make sure that the Property Tools components are properly installed on the user's machine.

First, the control must be registered. This is usually done by the installation program you use. If you need to register a component manually, you can use the command line regsvr32.exe. These programs can usually be found in the Visual Basic setup kit directory. To use them, bring up a command line prompt in the same directory as the component and the regsvr32 program. Run regsvr32 with the name of the component as the command line parameter. You can also unregister components by using the '-u' flag before the name of the component. In either case, you will get a message box telling you if the action was successful or not.

If you are distributing the File Property Component, then you should ship OLE32.DLL version 4.0 or later. Windows 2000, Windows NT 4, Windows 98 and Internet Explorer 4 (or later) all come with version 4.0 or later of this library.

You can locate the version number of a file by looking at the file's properties in the Windows Explorer program.

The Desaware File Property Component

Since the introduction of the structured storage format, many programs use it as the basis of the files they save to disk. Most applications, however, still use their own proprietary format. This is a problem because one of the key features of structured storage files – the ability to have a commonly readable table of file properties – is most useful if all documents have them. Otherwise, searching for files with certain authors or word counts will miss all those files without these properties. In response to this, Microsoft has introduced the ability to add many of these file properties to any file located on a Windows 2000 NTFS formatted drive. The Desaware Property control is designed to let you read and write file properties for files on such a drive, as well as structured storage files like Office documents on any drive or operating system.

Technology of File Properties

Structured storage files consist of blocks of data (called streams) held in hierarchical directories (called storages). At first, a single stream called "SummaryInformation" held file properties. Because this information was only being used by Microsoft Word at the time, it only held properties that related to word processor documents. As other Microsoft Office applications started using the structured storage format, these were no longer sufficient. Microsoft added a second stream, called "DocumentSummaryInformation", which added new properties specific to other Office products and a section for user-defined properties. The Desaware File Property Component, like the Desaware Storage Component, can read and write any of these properties, and can add them to any structured storage file that does not have them.

Every implementation of the NTFS disk format has supported multiple "streams" (not to be confused with structured storage streams) of information per file, although this is a little known or used feature. A NTFS stream is another section of the file, but it is not shown in the size of the file in Explorer, nor is it seen if you look at a file in a text or binary editor. You can experiment with streams on a command line. The following line will copy a list of files in the current directory into a new text file entitled "directory.txt".

```
dir > directory.txt
```

The next line will do the same, but the list of files will go into the stream named "section2"

```
dir > directory.txt:section2
```

Note how the file has not grown in size (at least according to the `dir` command or Explorer), and how Notepad only shows the single list. You can see what is in the "`section2`" stream by using the following command line:

```
more < directory.txt:section2
```

Windows 2000 uses NTFS streams to save file properties for any file, so setting properties for a file does not change its internal data. However, note that not all Microsoft operating systems support streams. If you copy a file to a non-NTFS drive you will lose the file properties. If you use an operating system that does not know about streams, like Windows 95, to copy a file on an NTFS drive, you will also lose any streams, even if the destination drive is also NTFS formatted. Windows 2000 only stores those properties found in the SummaryInformation stream, so you cannot add custom properties to non-structured storage files on NTFS drives. For more information on NTFS file streams, see article Q105763 in the Microsoft Developer's Network CDROM or on the Microsoft web site.

File Property Component Example

The following is an example of how you would use the File Property Component to read and write file properties:

```
' Make sure that "Desaware Property component"
' is selected in the Visual Basic references dialog.
' Create an instance of the Property component.
Dim dwProp As New dwProperty

' This determines if the file is a structured
' storage file or not.
IsStorage = dwProp.IsStorageFile ("C:\file.txt")

' Open a file.  This should be a structured storage
' file, or a file on a Windows 2000 NTFS drive.  If
' it is not one of these, you will get a runtime
' error.
dwProp.OpenFile "C:\file.txt", STG_FLAGS_NORMAL

' Read the summary information from the file.
If (dwProp.ReadSummaryInfo = True) Then
    ' get the title of the document
    titleString = dwProp.GetTitle()
    ' add something to the string
    titleString = titleString & " version 2"
    ' set the new title
    dwProp.SetTitle titleString
    ' save the changes to the
    ' SummaryInformation stream
    dwProp.SaveSummaryInfo
    ' when done making changes, release the file.
    dwProp.ReleaseFile
End If
```

File Property Component Properties

*ComponentObject.**Filename***

This property holds the name of the file that is currently open. If no file is currently open, it is blank.

File Property Component Methods

*ComponentObject.**OpenFile***
(Filename as String, Flags as Long)

You must first open a file before you can read or write properties to it. The **Filename** parameter is the path and name of the file to be opened. The **Flags** parameter describes the way the file is opened. Usually the constant STG_NORMAL is used here – see the section on constants for more information. When OpenFile is called on a normal file on an Windows 2000 NTFS drive, it does not keep the file open. If the file is opened or written to by another application after OpenFile is called and before other Property control methods are called no problem will occur. If the file is moved or deleted after OpenFile is called, you will get an error on calling ReadSummaryInformation or SaveSummaryInformation.

*ComponentObject.**ReleaseFile ()***

When you are done accessing a file, call the **ReleaseFile** method. This will clear certain internal buffers, and make sure the file is written to disk.

*ComponentObject.**ReadSummaryInfo () as Long***

Retrieves information from the SummaryInformation property set stream in the root storage of this Compound File. If there is no SummaryInformation property set, **ReadSummaryInfo()** generates an error. Before the **ReadSummaryInfo()** is called, using any function beginning with "Get" will return the default value, which will be either zero or a blank string depending on the type. The return value is a bitmap of the streams which the file contains.

Comparing the return value to the constants SUM_INFO, DOC_INFO and USR_INFO will tell you if the file contains a SummaryInformation stream, a DocumentSummaryInformation stream, and User-defined Summary Information respectively. Here is a sample of how to use this feature:

```
' Using the And operator will return a non-zero
' value if there are bits in the return value and
' bits in DOC_INFO that match.  See the Visual
' Basic documentation for "And" for more information
retval = dwProp.ReadSummaryInfo()
If (retval And DOC_INFO) then
    ' yes, this file has Document summary info
    Debug.Print dwProp.GetCompany()
End If
```

*ComponentObject.**SaveSummaryInfo ()***

Saves any changes to the SummaryInformation. If you have opened a structured storage file, these changes may not be written to disk until the **ReleaseFile** method is called.

*ComponentObject.**IsStorageFile***
(Filename as String) as Boolean

When passed a path and name of a file in the **Filename** parameter, this will return True if the file is a structured storage file, and False if it is not. Only compare the return value against False, as different programming languages have different values for True.

*ComponentObject.**EnableComponent (LicenseKey as String)***

This method allows you to use the File Property Component in your own Visual Basic components, such as ActiveX controls or ActiveX documents. Otherwise, the File Property Component will also think it is being used in Visual Basic design time, and will report a license error.

You first create a key by using the dwLicGen.exe program. This program will ask you to enter the compiled name of your component (not including a path, but including an extension). Each key is uniquely created from the component name. If you change your component name, you will have to create a new license key. After the program generates the key, you will have the option to copy it to the clipboard to post into your application.

You then call the **EnableComponent** method before you access any of the components methods, like in the Initialize or Sub Main functions. If you ever unload or remove the last reference to the File Property Component, you will have to go through this process again upon creating a new reference.

```
Dim dwProp As New dwProperty

Private Sub UserControl_Initialize()
   On Error GoTo initfailed
   ' Your key will be different.
   dwProp.EnableComponent "2E3499248322D3636853B53"
   Exit Sub
initfailed:
   ' Do appropriate error handling here.
End Sub
```

*ComponentObject.**IsValid () as Boolean***

Returns True if a file has been opened, False if not. Only compare the return value against False, as different programming languages have different values for True.

*ComponentObject.**ConvertToLocalTime (InputTime as Date) as Date***

This will convert a date from Greenwich Mean Time (**GMT**) to local time. Times in SummaryInformation properties are stored in **GMT** (this is done automatically by the properties of the component that use dates).

Summary Information Methods

The following methods all deal with the properties in the SummaryInformation stream. If the file does not have such a stream when it is opened, one will be created on calling SaveSummaryInformation.

*ComponentObject.**SetTitle (Title as String)***

Sets the title of the document.

*ComponentObject.**GetTitle () as String***

Returns the title of the document.

*ComponentObject.**SetSubject (Subject as String)***

Sets the subject of the document.

*ComponentObject.**GetSubject () as String***

Returns the subject of the document.

*ComponentObject.**SetAuthor (Author as String)***

Sets the author of the document.

*ComponentObject.**GetAuthor () as String***

Returns the author of the document.

*ComponentObject.**SetKeywords (Keyword as String)***

Sets key words related to the subject of the document. These key words are typically used in search routines.

*ComponentObject.**GetKeywords () as String***

Returns key words relating to the subject of the document.

*ComponentObject.**SetComments (Comment as String)***

Sets the comment field. This area might be used for making notes to oneself or others.

*ComponentObject.**GetComments () as String***

Returns the comment field.

*ComponentObject.**SetLastAuthor (LastAuthor as String)***

Sets the latest person to have modified the document.

*ComponentObject.**GetLastAuthor () as String***

Returns the last person who has modified the document.

*ComponentObject.**IncrementRevNum ()***

Increments the number of times a document has been revised by one.

*ComponentObject.**SetRevNum (TimesRevised as Long)***

Sets the number of times the document has been revised.

*ComponentObject.**GetRevNum () as Long***

Returns the number of times the document has been revised.

*ComponentObject.**StartEditTimer ()***

This starts a timer within the dwProp component. It is useful for keeping track how long a document has been opened, so you can update the TotalEditTime property. You can call this method when the file is first opened, and the **AddEditTimerToTotal** method when the file is saved.

*ComponentObject.**AddEditTimerToTotal ()***

Adds the amount of time since **StartEditTimer** was called to the property recording the total amount of time the document has been opened.

*ComponentObject.**GetTotalEditTime () as Long***

Returns the total amount of time a document has been open, in minutes.

*ComponentObject.**SetTotalEditTime (Minutes as Long)***

Sets the total amount of time a document has been open, in minutes.

*ComponentObject.**RecordPrintDate ()***

Sets the field containing the last time this document was printed to the current date and time.

*ComponentObject.**GetLastPrintDate () as Date***

Returns the last time the document was printed.

*ComponentObject.**RecordCreateDate ()***

Sets the field containing the time the document was created to the current date and time.

*ComponentObject.**GetCreateDate () as Date***

Returns the time the document was created.

*ComponentObject.**RecordSaveDate ()***

Sets the field containing the last time this document was saved to the current date and time.

*ComponentObject.**GetLastSaveDate () as Date***

Returns the last time the document was saved.

*ComponentObject.**SetNumberOfPages***
(NumPages as Long)

Sets the number of pages.

*ComponentObject.**GetNumberOfPages () as Long***

Returns the number of pages.

*ComponentObject.**SetNumberOfWords***
(NumWords as Long)

Sets the number of words.

*ComponentObject.**GetNumberOfWords () as Long***

Returns the number of words.

*ComponentObject.**SetNumberOfCharacters***
(NumChars as Long)

Sets the number of characters.

*ComponentObject.**GetNumberOfCharacters () as Long***

Returns the number of characters.

*ComponentObject.**SetApplication***
(AppName as String)

Sets the name of the program that created the document.

*ComponentObject.**GetApplication () as Long***

Returns the title of the program that created the document.

*ComponentObject.**SetTemplate***
(Template as String)

Sets the filename of the template used in the document.

*ComponentObject.**GetTemplate () as String***

Returns the filename of the template used in the document.

*ComponentObject.**SetSecurity***
(SecurityLevel as Long)

Sets the recommended security level of the document. Note that this does not actually supply any security, but only serves as a reminder to the application that reads the file.

Value	Security Procedure
0	None.
1	Password protected. Do not allow viewing or editing of this document. Other programs cannot view or edit this document.
2	Read-Only recommend. The user will be warned if there is any attempt to edit the document.
3	Read-Only enforced. Does not allow any changes to the document.
4	Locked for Annotations. Does not allow any changes to the document.

*ComponentObject.**GetSecurity () as Long***

Returns the security level of the document. Your program should behave as follows:

Value	Security Procedure
0	None.
1	Password protected. Do not allow viewing or editing of this document unless you know what type of password protection is involved, and the password given is correct.
2	Read-Only recommend. Warn the user if there is any attempt to edit the document.
3	Read-Only enforced. Do not allow any changes to the document.
4	Locked for Annotations. Do not allow any changes to the document.

Document Summary Information Methods

These methods access the DocumentSummaryInformation stream. Not every structured storage file that contains a SummaryInformation stream contains a DocumentSummaryInformation stream. If you open a structured storage file that only has a Summary Information stream and you use any of the Document Summary Information methods that begin with "Set", a DocumentSummaryInformation stream will be added to the file. Normal files on a Windows 2000 NTFS drive do not have these properties – only the those which have Summary Information. If you open a normal file on a Windows 2000 NTFS drive and use a Document Summary Information method that begins with "Set", it will have no effect.

*ComponentObject.**SetScaleCrop***
(Scale as Boolean)

Set this to True if this document is supposed to be scaled to the dimensions of the window, False if it is to be cropped to the dimensions of the window.

*ComponentObject.**GetScaleCrop () as Boolean***

Returns True if this document is supposed to be scaled to the current dimensions of the window, False if it is to be cropped to the dimensions of the window. Only compare this to "False" in Visual Basic, as True can refer to any non-zero number.

*ComponentObject.**SetLinksUpToDate***
(Links as Boolean)

Set this to True if all the links are up to date. Links may refer to internal links in the document or links to web sites depending upon the application.

*ComponentObject.**GetLinksUpToDate () as Boolean***

Returns True if all the links are up to date. Only compare this to "False" in Visual Basic, as True can refer to any non-zero number

*ComponentObject.**SetCategory***
(Category as String)

Sets the category of the document.

*ComponentObject.**GetCategory () as String***

Returns the category of the document.

*ComponentObject.**SetPresentationTarget***
(Target as String)

Sets the presentation target of the document. Used in Microsoft PowerPoint.

*ComponentObject.**GetPresentationTarget () as String***

Returns the presentation target of the document.

*ComponentObject.**SetManager***
(Manager as String)

Sets the manager of the document writer.

*ComponentObject.**GetManager () as String***

Returns the manager of the document writer.

*ComponentObject.**SetCompany (Company as String)***

Sets the company with which the document is associated.

*ComponentObject.**GetCompany () as String***

Returns the company with which the document is associated.

*ComponentObject.**SetNumBytes (NumBytes as Long)***

Sets the size of the document in number of bytes. Usually refers to the section of the document edited by the user, not the entire size of the file.

*ComponentObject.**GetNumBytes () as Long***

Returns the size of the document in number of bytes.

*ComponentObject.**SetNumLines (NumLines as Long)***

Sets the number of lines in the document.

*ComponentObject.**GetNumLines () as Long***

Returns the number of lines in the document.

*ComponentObject.**SetNumParagraphs (NumParas as Long)***

Sets the number of paragraphs in the document.

*ComponentObject.**GetNumParagraphs () as Long***

Returns the number of paragraphs in the document.

*ComponentObject.**SetNumSlides (NumSlides as Long)***

Sets the number of slides in the document.

*ComponentObject.**GetNumSlides () as Long***

Returns the number of slides in the document.

*ComponentObject.**SetNumNotes (NumNotes as Long)***

Sets the number of notes in the document.

*ComponentObject.**GetNumNotes () as Long***

Returns the number of notes in the document.

*ComponentObject.**SetNumHiddenSlides (NumHiddenSlides as Long)***

Sets the number of hidden slides in the document.

*ComponentObject.**GetNumHiddenSlides () as Long***

Returns the number of hidden slides in the document.

*ComponentObject.**SetNumMMClips (NumClips as Long)***

Sets the number of multimedia clips in the document.

*ComponentObject.**GetNumMMClips () as Long***

Returns the number of multimedia clips in the document.

User-Defined Property Methods

These methods access the user-defined portion of the DocumentSummaryInformation stream. Not every structured storage file that contains a SummaryInformation stream contains a DocumentSummaryInformation stream. Normal files on a Windows 2000 NTFS drive do not have these properties.

Here is an example of how you would use these methods to read and print a list of existing properties:

```
Dim count as Long
Dim v as Variant
Dim i as Long

' if the user-defined part of the
' DocumentSummaryInformation stream
' exists, load its information
count = 0
If (dwProp.OpenSummaryInfo And USR_INFO) Then
    ' get the number of properties
    count = dwProp.UserCount
    ' print all the property names and data
    For i = 0 To count - 1
        Debug.Print "name:", dwProp.UserDirectory(i)
        Debug.Print "data:", dwProp.UserGet(i)
    Next i
```

The following is an example of how you would add a number and a string into the Document Summary Information property set:

```
    ' Add a property that contains a number
    v = 1234
    count = dwProp.UserAdd("NewEntry1", v)
    ' Add a property that contains a string
    v = "String Value"
    count = dwProp.UserAdd("NewEntry2", v)

    ' This saves all summary information.
    dwProp.SaveSummaryInfo
    ' make sure the summary information is
    ' actually written to disk
    dwProp.ReleaseFile
End If
```

*ComponentObject.**UserSet***
(PropertyID as Long, Name as String, Data as Variant)

Modifies the existing property specified by **PropertyID**. **Data** can be either an integer, a date, a string, or a boolean value. Regular integers are converted to long integers to match the existing standard. If **PropertyID** does not correspond to an existing property, an error is generated. Due to certain limitations in Windows, dates must be between the years 1980 and 2107. If you need to store a date outside this range, you can use a string variant.

*ComponentObject.**UserGet***
(PropertyID as Long) as Variant

Returns the data associated with the property specified by **PropertyID**.

*ComponentObject.**UserAdd***
(Name as String, Data as Variant) as Long

Add a new property to the end of the list of user-defined properties in this Document Summary Information stream. The new count of user-defined properties is returned. Due to certain limitations in Windows, dates must be between the years 1980 and 2107 – if you have to store a date that is outside this range, you can use a string.

```
' Add a string
Dim v as Variant
v = "New String Data"
count = dwProp.UserAdd ("value name", v)
```

*ComponentObject.**UserCount () as Long***

Returns the number of user-defined properties in this Document Summary Information stream.

*ComponentObject.**UserDelete***
(PropertyID as Long) as Long

This deletes the specified property. After the deletion, all properties are renumbered to remove the empty space. The new count of user-defined properties is returned.

```
' Delete the last property
count = dwProp.UserCount
count = dwProp.UserDelete (count - 1)
```

*ComponentObject.**UserDirectory***
(PropertyID as Long) as String

Returns the name of the specified property.

File Property Component Constants

OpenFile flags parameter

any one of:

STG_READ = &H0

Read from, but not write to the file. Methods beginning with "Set" will still work, but an attempt to call **SaveSummaryInformation** will result in an error. With STG_DIRECT set, this is faster and takes less memory and disk space using other flag combinations.

STG_WRITE = &H1

Write to, but not read from the element.

STG_READWRITE =&H2

Allows both reading from and writing to the element.

any one of:

STG_SHARE_DENY_READ = &H30

Does not allow any other to read from this element.

STG_SHARE_DENY_WRITE = &H20

Does not allow any other to write to this element.

STG_SHARE_EXCLUSIVE = &H10

Does not allow any other attempt to access this file in any way from the time it is opened to when the **ReleaseFile** method is called. If the file is a normal file on a Windows 2000 NTFS drive, it must have this flag selected; however, such files are only accessed at the time of reading or writing.

STG_SHARE_DENY_NONE = &H40

Allows any other to read from or write to this file as long as it is open.

either one of:

STG_DIRECT = &H0

All changes are made in a more direct fashion (although changes are not always written directly to the file, depending on the version of OLE installed). This is faster and requires less memory. If the file is a normal file on a Windows 2000 NTFS drive, it must have this flag selected

STG_TRANSACTED = &H10000

All changes are made to an internally kept copy of the storage. This is more memory intensive, but allows a greater array of flag choices. It also provides other advantages that do not relate to summary information sets.

Or you can just use:

STG_NORMAL

STG_NORMAL is a combination of STG_SHARE_ EXCLUSIVE, STG_DIRECT, and STG_READWRITE. This is the most common flag combination, and will let you access any file that is not already open.

List of all possible flag combinations:

```
STG_DIRECT or STG_SHARE_EXCLUSIVE or STG_READWRITE
STG_DIRECT or STG_SHARE_EXCLUSIVE or STG_WRITE
STG_DIRECT or STG_SHARE_EXCLUSIVE or STG_READ
STG_TRANSACTED or STG_SHARE_EXCLUSIVE or STG_READWRITE
STG_TRANSACTED or STG_SHARE_EXCLUSIVE or STG_WRITE
STG_TRANSACTED or STG_SHARE_EXCLUSIVE or STG_READ
STG_TRANSACTED or STG_DENY_WRITE or STG_READWRITE
STG_TRANSACTED or STG_DENY_WRITE or STG_WRITE
STG_TRANSACTED or STG_DENY_WRITE or STG_READ
STG_TRANSACTED or STG_DENY_READ or STG_READWRITE
STG_TRANSACTED or STG_DENY_READ or STG_WRITE
STG_TRANSACTED or STG_DENY_READ or STG_READ
STG_TRANSACTED or STG_DENY_NONE or STG_READWRITE
STG_TRANSACTED or STG_DENY_NONE or STG_WRITE
STG_TRANSACTED or STG_DENY_NONE or STG_READ
```

Index

Other Sources of Information

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

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.

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.

Windows API Online Help

The Professional Edition of Visual Basic includes Win31api.hlp and/or win32api.hlp - an online help reference for all API functions. These functions are declared in C and do not consider Visual Basic compatibility issues, however the information in chapter 3 of the Visual Basic Programmer's Guide to the Windows API (chapters 3 and 4 of the 32 bit book) will provide you with information on how to translate these functions to Visual Basic.

Microsoft's Developers Network CD Rom

This amazing CD-ROM contains a wealth of information and sample code, plus the latest Visual Basic knowledge base.

Microsoft's Windows Software Development Kit and Win32 Software Development Kit

The sample code is all in C, but by the time you've read the Visual Basic Programmer's Guide to the Windows API or Win32 API, you'll know enough to be able to translate the C code to Visual Basic.

Desaware Product Descriptions

Thank you for your purchase of this Desaware product. We have additional quality software to enhance your programming efforts. Please visit our web site at **www.desaware.com** for detailed descriptions and product demos.

SPYWORKS 6

SpyWorks in a nutshell? Impossible!

You're going to want to download the SpyWorks demo to even begin to understand its capabilities. This product has been evolving for several years, and it includes so many features it's hard to know where to begin. SpyWorks is a VB power tool. When you need to override VB's default behavior or to extend VB's functionality, you will want to use SpyWorks.

Do *That* in Visual Basic??

Want to put VB to the test? Want to learn advanced programming techniques? Want to keep the productivity of VB and have the functionality of C++? SpyWorks contains the low level tools that you need to take full advantage of Windows. Here are just a few of the features of this multi-faceted software package. For instance, have you ever wanted to detect keystrokes on a system-wide basis or detect when an event occurs in another application or thread using subclassing or hooks? SpyWorks can help you solve these problems by letting you tap into the full power of the Windows API without having to be an expert. SpyWorks lets you export functions from VB DLL's so that you can create function libraries, control panel applets, and NT Services. With its ActiveX extension technology, you can call and implement interfaces that VB5 or 6 do not support. SpyWorks includes the Desaware API Class Library which assists programmers in taking advantage of the hundreds of functions that are built into the Windows API. SpyWorks is available in either the Professional (Pro) or Standard edition.

The Pro Edition includes a WinSock component with comprehensive VB source code that gives you complete control for Internet/intranet programming. Other features are the NT Service launcher. This new application is a subset of the Desaware NT Service Toolkit product. It allows a developer to communicate with an NT service through VB created objects. A background thread component. Multithread your VB executable. Allows you to easily create objects that run in a separate background thread.

It also contains extensive sample code and three product updates.

- 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.

- SpyWorks Standard is a subset of Professional. A feature comparison is available on our web site.

- Supports VB 4, 5 & 6, Windows 95, 98, 2000 & NT.

VERSIONSTAMPER 6.0

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!

- Checks the versions of your dependent files and notifies you or the user of potential problems.
- Internet extensions allow you to update versions across the Internet/intranets.
- 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

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 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.