Desaware Event Log Toolkit™

Version 1.0

for Visual Basic

by

Desaware, Inc.

Rev: 1.0.1 (06/2005)

Desaware Inc.
3510 Charter Park Drive, Suite 48
San Jose, CA 95136
(408) 404-4760

www.desaware.com

Copyright ©2000-2005 by Desaware Inc. All Rights Reserved

Information in this document is subject to change without notice and does not represent a commitment on the part of Desaware, Inc. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Desaware, Inc.

License Agreement & Warranty

Desaware, Inc.
Software License

Please read this agreement. If you do not agree to the terms of this license, promptly return the product and all accompanying items to the place from which you obtained them.

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

This program will be licensed for use on a single computer. If you wish to transfer the license from one computer to another, you must uninstall it from one computer before installing it on the next. 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, 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. Companies or schools interested in multiple copy licenses or site licenses should contact Desaware, Inc. directly at (408) 404-4760.

Should your intent be to purchase this product for use in developing a compiled Visual Basic program that you will distribute as an executable (.exe or .dll) file, review the listing of which files (located below and in the File Description section of the product manual) can be distributed and or modified. If Desaware files are included in your executable program, you must include a valid copyright notice on all copies of the program. This can be either your own copyright notice, or "Copyright © 2000-2005 Desaware, Inc. All rights reserved.".

Files: You may distribute event source DLL files created using the Desaware event source utility. You may **not** modify the files listed above in any way.

Source Files: Source code for portions of the Desaware EventLog Toolkit are included for educational purposes only. You may use this source code in your own applications only if they provide primary and significant functionality beyond that included in the toolkit package. You may not use this source code to develop or distribute components and tools that provide functionality similar to all or part of the functionality provided by any of the components or tools included in the Event Log Toolkit package.

Please consult the on-line Help file under the topic File Descriptions for additional information.

Microsoft is a registered trademark of Microsoft Corporation. Visual Basic, Windows, Windows 95 and Windows 98 are trademarks of Microsoft Corporation.

Desaware NT Service Toolkit, Desaware Event Log Toolkit, SpyWorks, VersionStamper, StorageTools, ActiveX Gallimaufry, Custom Control Factory, and SpyNotes #2, The Common Dialog Toolkit are trademarks of Desaware, Inc.

Limited Warranty

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

The entire and exclusive liability and remedy for breach of this Limited Warranty shall be limited to replacement of defective diskette(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., 3510 Charter Park Drive, Suite 48, San Jose, California 95136.

Table of Contents

Introduction

The NT Event Log is the preferred way for services and server components to log errors, warnings and other information. Typical uses of the event log include:

- Reporting information in cases where it is not possible or advisable to bring up a message to the current logged on user.
- Reporting information that does not require immediate response.
- Reporting information that you wish to archive for later evaluation or for auditing purposes.

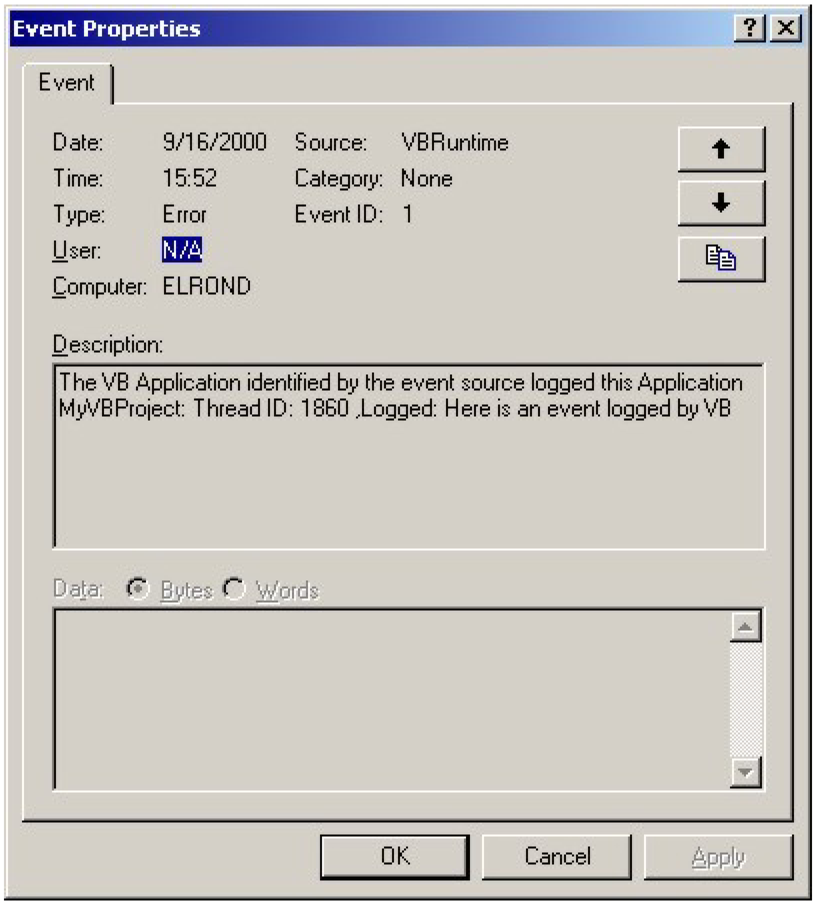
The event log allows you to classify events in several ways, by severity of the error, by source of the error, and by categories that you define. Event viewing tools can then sort or filter event information based on these classifications. The event log system also makes it easy to define events in a manner that is language independent – so events show up correctly in the language of the local system.

The Wrong Way to Use the Event Log with Visual Basic

When Visual Basic added the App.LogEvent method, VB programmers were thrilled to have the ability to easily log events to the event log. The App.LogEvent method could be called easily as shown in this example called from a VB application called "MyVBProject:

```
App.LogEvent "Here is an event logged by VB",
vbLogEventTypeError
```

Figure 1 depicts the event in the event log entry that results from this call:



Date: 9/16/2000 Source: VBRuntime
Time: 15:52 Category: None
Type: Error Event ID: 1
User: N/A
Computer: ELROND

Description:

The VB Application identified by the event source logged this Application MyVBProject: Thread ID: 1860 ,Logged: Here is an event logged by VB

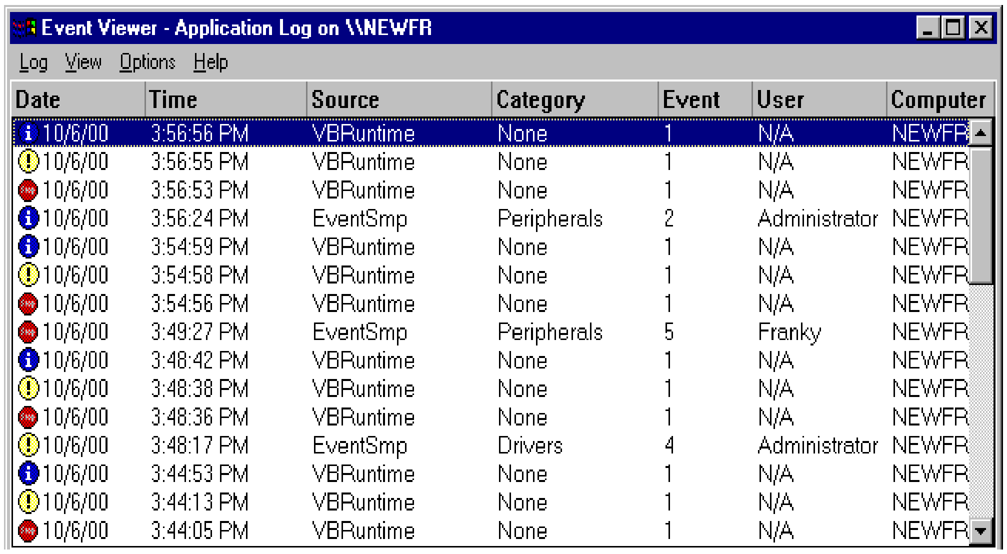
Figure 1
Typical VB Event Resulting From a Call to the App.LogEvent Method

What's wrong with this picture?

- The Source is VBRuntime – not MyVBProject. So it is impossible to distinguish between events logged by your application and those by any other VB application (See Figure 2).
- The event description is the same for all VB applications. You have to read the description to find the application name and the string used in the LogEvent method.

- The string used in the LogEvent application is language dependent – it will always appear in the event log in the language in which it was written.
- There is no ability to define or report categories, making it impossible to classify event.

In other words, it is impossible to use the event log correctly with the App.LogEvent method.



Event Viewer - Application Log on \\NEWFR

Log View Options Help

Date	Time	Source	Category	Event	User	Computer
10/6/00	3:56:56 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:56:55 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:56:53 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:56:24 PM	EventSmp	Peripherals	2	Administrator	NEWFR
10/6/00	3:54:59 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:54:58 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:54:56 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:49:27 PM	EventSmp	Peripherals	5	Franky	NEWFR
10/6/00	3:48:42 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:48:38 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:48:36 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:48:17 PM	EventSmp	Drivers	4	Administrator	NEWFR
10/6/00	3:44:53 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:44:13 PM	VBRuntime	None	1	N/A	NEWFR
10/6/00	3:44:05 PM	VBRuntime	None	1	N/A	NEWFR

Figure 2
Event Log Viewer

This image of the Event Log viewer shows that it is impossible to distinguish between events logged by two different VB projects using the App.LogEvent method. Note how VB projects that use event sources (EventSmp) created with this Toolkit can be easily identified.

What About Event Log API Functions?

The solution is to use either the event log API functions, or components that wrap those functions. However, as you will soon see, just calling the API functions is not enough. In order to create custom events, you need the ability to create an event source – a special kind of file that contains message and category definitions in the languages that you wish to support. Creating event sources has always been rather complicated and poorly documented. It has also been historically difficult to distribute event sources, as they require specialized registry entries in order to work correctly.

The Correct Way to Use the Event Log from Visual Basic

The Desaware Event Log Toolkit is the first product that not only makes it easy to log events from Visual Basic, it makes it easy to create and distribute custom event sources. In fact, it's so easy to create event sources with our toolkit that even Visual C++ programmers will find it a superior approach to Microsoft's tools.

The Desaware Event Log Toolkit supports the following features:

- Create custom event sources with an interactive Windows application – no complex file formats to learn and edit.
- Event sources support unlimited languages.
- Define custom categories for your event sources.
- Messages are defined by combining identifiers, facilities, and severity information as recommended by Microsoft (you'll read more about this later).
- Automatic generation of VB module and C++ header files with constant definitions.
- Event sources are self-registering – all necessary registry entries can be added automatically using regsvr32.
- Event sources have no component or DLL dependencies – just ship and register.
- Event log class allows access to the event log API – all VB source code included: just drop the class into your project and use it.

- Event log class allows reporting events with all possible parameters – including the difficult to implement ability to specify user accounts with events.

In order to understand how to use the event log, your first step should be to understand exactly what the event log is, and why it works the way it does.

Inside the Windows NT/2000 Event Log

To understand the rather convoluted architecture of the NT/2000 event log, it is necessary to understand the purpose behind its design. First and foremost, the event log was designed to log information produced by system components such as drivers and services. That means that much of the information would have one or more of the following characteristics:

- It would be generated before a user logged on to the system – meaning that notifying a user through a message box was not an alternative.
- It would contain information that was not of immediate interest to a user, but that could be useful to system administrators in evaluating the performance of a system or diagnosing system errors – including errors that did not prevent the system from being used (for example: those that impacted only one subsystem).
- It would contain information that needs to be localized. Since Windows itself is localized to many different languages, and many events are generated by Windows operating system components, localization became an important design factor from the beginning. What's more – it was important to be able to add new languages without modifying each of the system components to support the new language, or having a different version of system components for each language.

To address these issues, Microsoft adopted the event log architecture shown in Figure 3.

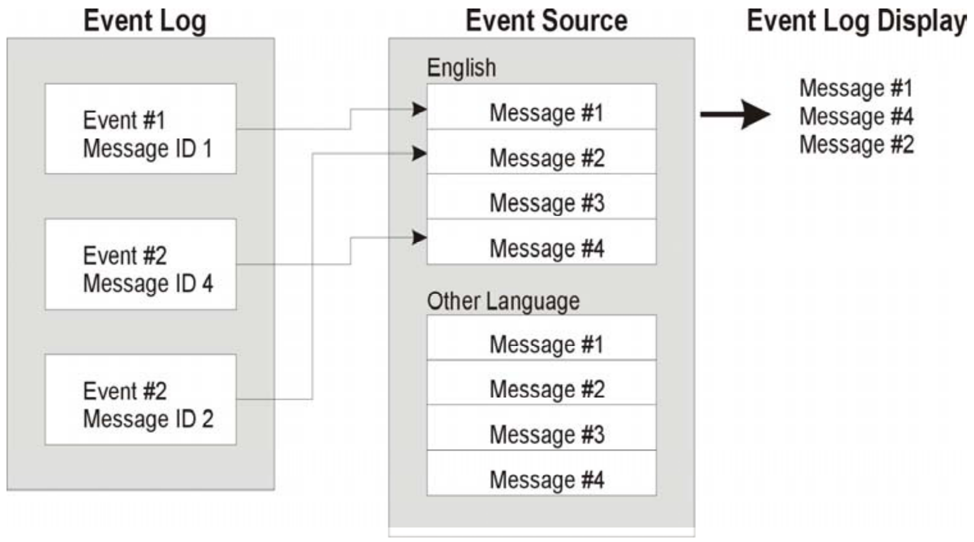


Figure 3
Event Log Architecture

The event log itself does not contain any text that needs to be localized. Instead, it contains information identifying an event source, an event identifier, and any language independent text or binary data that it wishes to report. The event source contains the actual message text for every language supported by the event source. Thus, if you support English, French and Japanese, you would find message #1 available in the event source in all three languages.

When the event log is displayed, the event log viewer reads this information from the event log. It opens the event source and looks up the message identifier for the current language being used by the system (the current locale), then displays the correct text. The event log viewer can also merge language independent text in the event log into the messages (you'll read more about this next).

Inside a Message

Because the event log was designed to log events to be viewed at a later time, it was also designed to make it easy to categorize and filter events. And while it is easy for system managers to filter and search for events with the event log viewer, the way those categories are defined from a developer's perspective are less clear (to put it kindly).

When you log an event, the parameters to the ReportEvent method include the following:

- The event source
- The event category
- The severity of the event (error, warning, information, auditing).
- The event identifier.
- The user that logged the event (optional)
- Other language independent text
- Binary data.

Let's review these one at a time.

The Event Source

Each event source has a name and is registered in the system under the registry key

```
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/ _
   Services/EventLog
```

There are three subkeys under this key: *Application*, *Security* and *System*.

The name of the event source appears as a subkey under these three keys, thus if you create an event source named MyEventSource, it would appear in the registry under:

```
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/ _
   Services/EventLog/Application/MyEventSource
```

An event will appear in the event log under the Application, Security or System event logs depending on where the event source is registered.

The Desaware Event Log Toolkit always registers event sources under the Application key, because the vast majority of people using the toolkit will be creating event sources for applications, components and services – all of whose events should appear in the Application event log. The System event log is intended for system components and drivers. The Security event log is intended for security audit information.

The MyEventSource registry key contains multiple named values.

Required registry values:

EventMessageFile	The full path to the event source file.
TypesSupported	Severity types supported by this event source.

Optional registry values:

CategoryMessageFile	The full path to the file containing message categories.
CategoryCount	The number of categories in the file.

The Desaware Event Log Toolkit automatically enters these values in the registry, including the category entries if categories are defined.

The automatic registration built into event log sources created by this Toolkit should suit the needs for virtually all situations. However, you always have the option of doing your own event source registration. Thus it is possible, for example, to manually register an event source created with the Desaware Event Log Toolkit for use with the system event log if you are creating a device driver.

The Event Category

An event source may define categories of events. These are always specific to an event source. Categories are numbered from 1 through the number of categories. Each one has a category name (which is also language independent – so each category has a text string for each supported language).

The Event Severity

Events fall into five possible types (or severities):

Error	Use to indicate a major failure in an application or component.
Warning	Use to indicate a condition that is not immediately fatal, but that could cause problems if ignored.
Information	Use to indicate significant events.
Success Audit	Use to log successful operations that are being audited by the security system.
Failure Audit	Use to log operations that fail due to security considerations.

Generally speaking, you will only use the first three event types, as security audit information is logged by the operating system depending on the security settings and system policies.

The Event Identifier

Here is where things begin to get tricky.

An event identifier is comprised of three different values that are combined using the logical OR operator – the event code, the facility, and the severity. These are laid out in the event identifier as shown here:

```
//  Values are 32 bit values laid out as follows:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//  +---+-+-+-----------------------+-------------------------------+
//  |Sev|C|R|     Facility          |               Code            |
//  +---+-+-+-----------------------+-------------------------------+
//
```

The severity value can be one of the following values

0 = Success
1 = Information
2 = Warning
3 = Error

The Facility values can be any values you wish to help you to categorize messages. Most will use the default facility code of &HFFF which corresponds to “Application”. Facility codes smaller than 256 are reserved by the system. The “C” and “R” bits are reserved and can be left as zero.

At this point you are probably very confused: If the severity is determined when reporting the event (as described earlier), and the event log (Application, System or Security) is determined by the location of the event source in the registry, what is the relation between the severity and log as reported, and the severity and facility in the event identifier?

There is no relation.

The severity and facility values in the event identifier are intended to help developers to manage events – especially when there are large numbers of events. Consider the following events that one might specify when creating a service.

```
    ServiceStarted = 1
    ServiceStopped = 2
    ServiceTCPError = 3
    ServiceDNSError = 4
    ServiceDied = 5
    TooManyClients = 6
```

Which of these are serious? Which of them relate to the service itself and which to network connectivity?

Now imagine that the events were reported as follows:

```
ServiceStarted =  SEVERITY_INFORMATION Or Application Or 1
ServiceStopped = SEVERITY_INFORMATION Or Application Or 2
ServiceTCPError =  SEVERITY_ERROR Or Network Or 3
ServiceDNSError = SEVERITY_ERROR Or Network Or 4
ServiceDied = SEVERITY_ERROR Or Application Or 5
TooManyClients = SEVERITY_WARNING Or Application Or 6
```

A programmer reading this code can instantly determine which events are severe, and whether events are associated with the service itself (Application) or a network operation.

Obviously, it is in your interest to use the same severity values in the event identifier as you do when reporting the event – doing otherwise could lead to confusion.

The Logging User

You can specify a user account when logging an event. The event logging classes included with the Desaware Event Log Toolkit allow you to do so. In most cases users information is not included in the event log (especially services, which typically run under the local system account).

Language Independent Text

The message text in the event source can contain parameters which allow you to merge in additional text provided when reporting the event. Text that you merge in using ReportEvent should be language independent (file names, for example).

The parameter locations are identified by escape sequences that are indicated by the % character. The following escape sequences are supported:

Escape Sequence	**Definition**
%0	Ends the text without a new line (crlf) character.
%n	Merges in string #n specified by the ReportEvent function. Thus if you pass three strings to ReportEvent, %2 will merge the second of these strings into the message text at the location of the %2.
%n!printformat!	Like %n, except that the data is formatted according to the print format characters. These characters are the same as those used by the *printf* command in the C language (refer to your online documentation or MSDN for a complete list of print format specifiers).
%%	Replaced by the '%' character.
%n	Add line break.
%r	Adds return (without a new line character).
%*space*	Adds a space character.

%.	Adds a period (without terminating the message).
%!	Adds an exclamation point.

Examples:

Message	**String Parameters**	**Result**
Operation %1	Succeeds	Operation Succeeds.
Operation %1 %n %2 %!	Succeeds, Reboot	Operation Succeeds Reboot!
Function %1 %% Complete	90	Function 90 % complete.

Binary Data

You can attach any arbitrary binary data to an event when reporting the event. The binary data will be displayed in hexadecimal format by the event viewer.

Using the Desaware Event Source Utility

The Desaware Event Source utility creates an event source file based on the event source name, messages, severity, categories, facilities and languages specified. The resulting event source file is compiled as a self-registering DLL file that does not require any additional dependency files.

Creating an Event Source File - Quick Start

To create an event source file, you need to specify the Application (or Event Source) name, version information for the event source, and at least one event message.

Select the Event Source Utility's **Edit** – **Version Resources** menu and enter the version resource strings for your compiled event source file. At a minimum, you must enter the File Version Number and Company Name. Select the **Ok** button when finished.

Select the Event Source utility's **Edit** – **Registry Settings** menu and enter the Application (event source) name. This will typically be the name of the application using the event source, but can be any name you wish. Select the **Ok** button when finished.

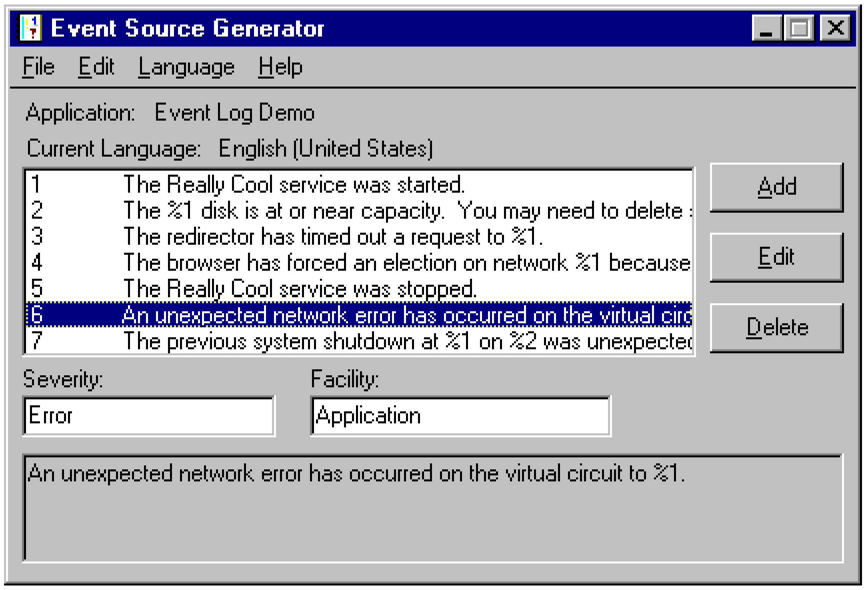
Select the **Add** button to add an event message. Select a Severity, Facility, and Message ID number for your message. Enter the message string, then select the **Ok** button when finished. Repeat this step until you have finished adding all the event messages you need.

You can save your event source project file before building the event source file by selecting the **File** – **Save As** menu command.

Select the **File** – **Build Source** menu to build the event source file. Enter the destination path and file name for the event source or select the **Browse** button to select the destination path and file name. Select the **Ok** button when finished.

Event Source Primary Form (Event Source Generator)

The primary event source utility form displays the current Application name for the event source, the currently selected Language for the event messages, and the current event messages. Selecting the **Add** button allows you to add a new event message for the selected language. The **Edit** button allows you to edit the selected event message for the selected Language. The **Delete** button deletes the selected event message for the selected language.



Application: Event Log Demo

Current Language: English (United States)

```
1    The Really Cool service was started.
2    The %1 disk is at or near capacity.  You may need to delete :
3    The redirector has timed out a request to %1.
4    The browser has forced an election on network %1 because
5    The Really Cool service was stopped.
6    An unexpected network error has occurred on the virtual circ
7    The previous system shutdown at %1 on %2 was unexpectec
```

Severity: Error

Facility: Application

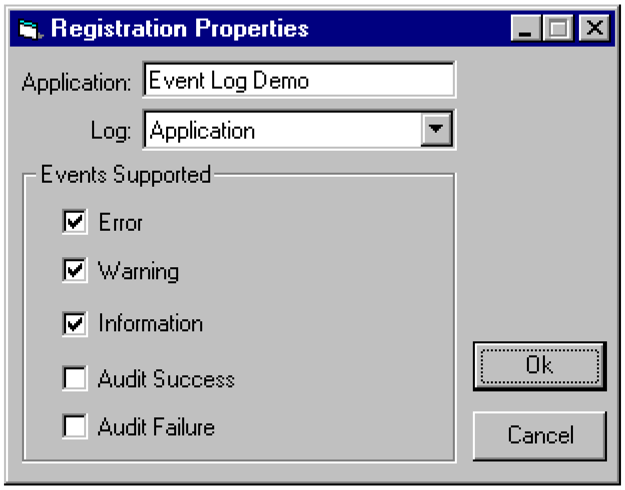
An unexpected network error has occurred on the virtual circuit to %1.

Figure 4
Event Source Generator
(the Primary Event Source Form)

Registry Settings Form

The Registry Settings form holds the information to be written into the registry when the event source file is registered. Enter the event source name in the Application text box. Select from Application, System, or Security as the Log type of your event source file (you should use the Application log for all applications and services).

Check the appropriate Events check boxes to select the types of severity your event source will support. Select the **Ok** button to save the changes, or the **Cancel** button to cancel the changes.


Registration Properties

Application: Event Log Demo

Log: Application

Events Supported

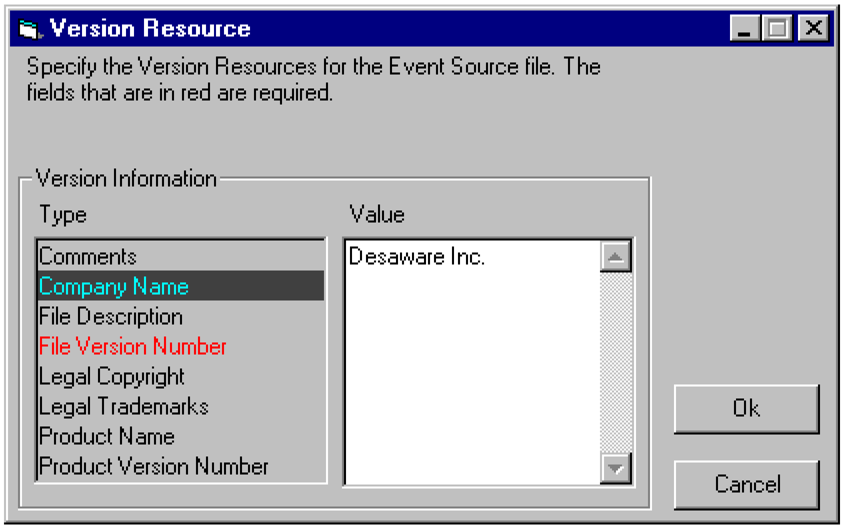
- [x] Error
- [x] Warning
- [x] Information
- [ ] Audit Success
- [ ] Audit Failure

Figure 5
Registration Properties Form

Version Resources Form

The Version Resources form is used to hold the version resources that are written to the event source file. The Company Name and File Version Number fields are required. All of the other version fields are optional but recommended. The File Version Number and Product Version Number fields should be formatted as “#.#.#.#” (for example 1.2.3.4).

Select an entry from the Version Resource Type list box, then enter the value for that field in the Version Resource Value text box. Select the **Ok** button to save the changes, or the **Cancel** button to cancel the changes.


Version Resource

Specify the Version Resources for the Event Source file. The fields that are in red are required.

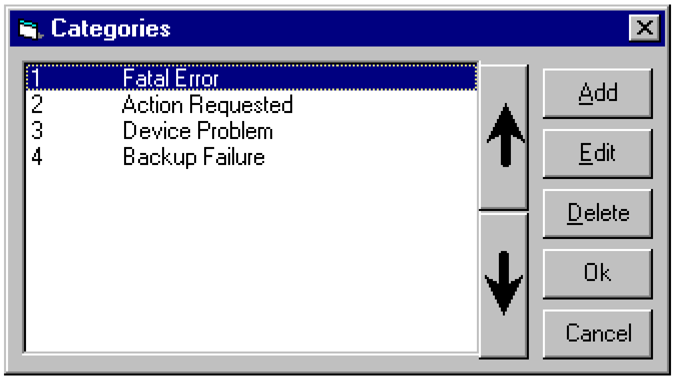
Version Information

Type	Value
Comments	Desaware Inc.
Company Name	
File Description	
File Version Number	
Legal Copyright	
Legal Trademarks	
Product Name	
Product Version Number	

Figure 6
Version Resource Form

Categories Form

The Categories form allows you to specify custom categories for your event source file. The list box displays the existing Categories. The Categories are automatically assigned a Category number when they are added. The Category numbers begin with 1 and increment by 1 for each additional category. Select a Category in the list box and the **Up** or **Down** arrow buttons to rearrange the Categories. Select the **Add** button to add a new Category to the list. The Category name is limited to 20 characters. Select the **Edit** button to change the name of the selected Category. Select the **Delete** button to delete the selected Category. Select the **Ok** button to save the changes, or the **Cancel** button to cancel the changes.



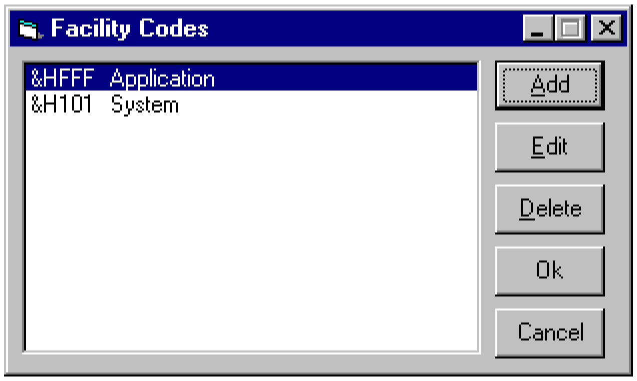
1	Fatal Error
2	Action Requested
3	Device Problem
4	Backup Failure

Figure 7
Categories Form

Facilities Form

The Facilities form allows you to specify custom facilities for your event source file. The list box displays the existing Facilities and their values. Select the **Add** button to add a new Facility to the list. Select the **Edit** button to change the name or value of the selected Facility. The Facility name is limited to 20 characters. The Facilities values must be between 256 (&H101) and 4095 (&HFFF). The value 4095 (&HFFF) is normally used for the Application Facility.

Select the **Delete** button to delete the selected Facility. Select the **Ok** button to save the changes, or the **Cancel** button to cancel the changes.



&HFFF Application
&H101 System

Figure 8
Facilities Language Form

Select Languages Form

The Select Languages form allows you to add or remove support for additional languages in your event source. Each language includes its own list of Event Messages, Categories, and Facilities.

Select the **Add** button to add a new Language to the list. When you add a new language, a copy of the existing Event Messages, Categories, and Facilities for the current language is made and added to the new language (you should then edit each of these entries, translating them into the new language). Select the **Delete** button to delete the selected language. When you delete a language, all of the language's Event Messages, Categories, and Facilities are deleted. Select the **Ok** button to close the form.

Tip: If you will be adding multiple language support, you should complete the Event Messages, Categories, and Facilities for one language first. Then you can add another language which will be created with a copy of the same Messages, Categories, and Facilities numbers as the first language.

Build Event Source Form

The Build Event Source form allows you to compile your Event Source file. It also includes options to export the defined Message, Category, and Facility numbers to a Visual Basic Module file or a Visual C++ header file.

Specify a destination path and file name for your event source file or use the **Browse** button to select a path and file name. Select the Generate Visual Basic Module File or Generate Visual C++ Include File check boxes to export the Message, Category, and Facility numbers. Similarly, specify a destination path and file name for the Visual Basic or Visual C++ output files or use the **Browse** button to select a path and file name. Select the **Ok** button to build the event source file and exported files. Select the **Cancel** button to cancel the build.

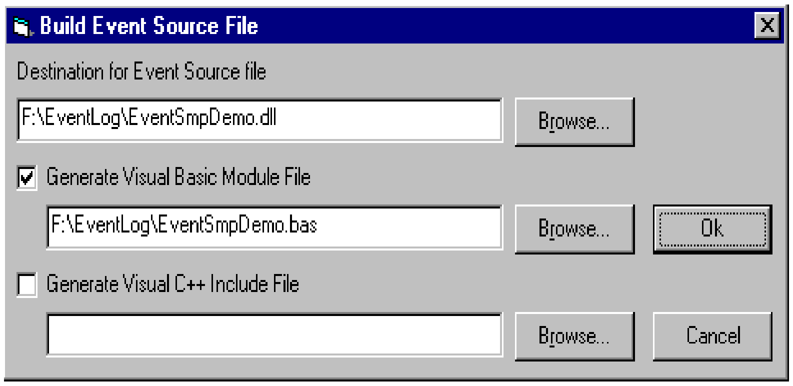

Build Event Source File
Destination for Event Source file
F:\EventLog\EventSmpDemo.dll
Browse...
Generate Visual Basic Module File
F:\EventLog\EventSmpDemo.bas
Browse...
Ok
Generate Visual C++ Include File
Browse...
Cancel

Figure 9
Build Event Source File

Event Source Project File

The information you enter using the event source utility can be saved into an event source project file. Event source project files can be opened, saved, and created using the event source utility.

Event Source Menu Commands

File

New	Creates a new event source project file.
Open	Opens an event source project file.
Save	Saves the current event source information into the event source project file.

Save As	Saves the current event source information into the specified event source project file.
Build Source	Displays the build event source form.
Exit	Closes the event source utility.

Edit

Version Resource	Displays the version resources form.
Registry Settings	Displays the registry settings form.
Categories	Displays the categories form.
Facilities	Displays the facilities form.
Languages	Displays the languages form.

Language

Contains one or more languages for the current event source. Select a language from this list to select the current working language for the event source.

Help

Help Topics	Displays the Desaware Event Log Toolkit help file.
About	Displays version information for the event source utility.

The Desaware Event Log API Class

The Event Log API functions can be tricky to call from Visual Basic. Therefore the Desaware Event Log Toolkit includes the dwEventLog class which provides wrappers for most of the event log API functions. This class is provided as source code so you can simply add it to any project.

To use the class, simply create a new object of type dwEventLog thus:

```
Dim el as New dwEventLog
```

The dwEventLogTypes Enumeration

The dwEventLogTypes public enumerated value is used to specify the severity of an event.

```
Public Enum dwEventLogTypes
   EVENTLOG_ERROR_TYPE = &H1
   EVENTLOG_WARNING_TYPE = &H2
   EVENTLOG_INFORMATION_TYPE = &H4
   EVENTLOG_AUDIT_SUCCESS = &H8
   EVENTLOG_AUDIT_FAILURE = &H10
End Enum
```

ReportEvent

Use this method to log an event into the event log.

```
Public Sub ReportEvent(ByVal Source As _
   String, ByVal SourceMachine As String, _
   ByVal EventType As dwEventLogTypes, ByVal _
   Category As Integer, ByVal EventID As _
   Long, Optional MergeStringsArray As _
   Variant, Optional BinaryDataArray As _
   Variant, Optional UserName As String, _
   Optional UserMachine As String)
```

The parameters are as follows:

Parameter	**Definition**
NameSource	The name of the event source.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.

EventType	The severity of the event. Select from the dwEventLogTypes enumerator.
Category	The category number in the source, zero for no category.
EventID	The event identifier.
MergeStringsArray	A zero based variant containing an array of strings (strings are numbered from 1 to N – leave array index zero empty).
BinaryDataArray	A variant containing a byte array (from 0 to N – position 0 in the array is valid).
UserName	The name of the user to associate with the event.
UserMachine	The server on which the user is valid.

GetNumberOfEventLogRecords

Retrieve the number of records in the specified event log.

```
Public Function GetNumberOfEventLogRecords _
   (ByVal Source As String, ByVal _
   SourceMachine As String) As Long
```

The parameters are as follows:

Parameter	Definition
Source	The name of the event source whose corresponding event log is to be read.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.

GetOldestEventLogRecord

Retrieves the record number of the oldest record in the event log.

```
Public Function GetOldestEventLogRecord _
   (ByVal Source As String, ByVal _
   SourceMachine As String) As Long
```

The parameters are as follows:

Parameter	Definition
Source	The name of the event source whose corresponding event log is to be read.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.

ReadEventLog

This method reads an entry in the event log into the dwEventLog object.

```
Public Sub ReadEventLog(ByVal Source As _
   String, ByVal SourceMachine As String, _
   ByVal EventIndex As Long)
```

The parameters are as follows:

Parameter	Definition
Source	The name of the event source whose corresponding event log is to be read.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.
EventIndex	The number of the event in the event log (starting from zero).

Once an event has been loaded into the object using this method, you can retrieve information about the event using the following methods:

EventSource

The event source for this event.

EventComputer

The computer containing the event source for this event.

EventID
The event identifier for the event.

EventString
The formatted message for the event (with parameter strings merged, as it would be displayed by the event log viewer).

EventCategory
The category number for this event.

EventCategoryString
The name of the category for this event.

EventType
The severity of the event.

EventBinary
Any binary data associated with this event.

EventUser
The user (if any) that recorded this event.

EventDomain
The domain for the user that recorded this event.

EventGenerated
The date and time at which the event occurred.

EventWritten
The date and time at which the event was recorded in the event log.

EventRecordNumber
The absolute record number of this event.

InsertionStringCount
The number of insertion strings found for this event.

InsertionString(ByVal stringindex As Long)

The insertion string as specified by stringindex. Stringindex begins at zero.

WasEventSourceFound

True if the Event Source file was found, False otherwise.

Note: The ReadEventLog function ReadEventLog uses the EVENTLOG_SEQUENTIAL_READ flag to move to the specified EventIndex to read. This is due to a Microsoft problem where large event log files (Microsoft mentions 2MB but our testing showed that this fails on smaller log files), the EVENTLOG_SEEK_READ method fails.

We have found that the speed differences between the two methods are insignificant. Also note that this function is not optimized for reading all records in a log file since it opens and closes the log file for each read operation.

BackupEventLog

Backs up an event log into a file.

```
Public Sub BackupEventLog(ByVal Source As _
   String, ByVal SourceMachine As String, _
   ByVal BackupFileName As String)
```

The parameters are as follows:

Parameter	**Definition**
Source	The name of the event source whose corresponding event log is to be backed up.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.
BackupFileName	The full path and name of the file in which to save the event log. This function will fail if the file already exists.

ClearEventLog

Clears the specified event log.

```
Public Sub ClearEventLog(ByVal Source As _
   String, ByVal SourceMachine As String, _
   ByVal BackupFileName As String)
```

The parameters are as follows:

Parameter	Definition
Source	The name of the event source whose corresponding event log is to be cleared.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.
BackupFileName	The full path and name of the file in which to save the event log before clearing it.

IsEventLogFull

Determines if the event log is full. This function is valid in Windows 2000 or later only. An error will be raised if this function is called outside of Windows 2000.

```
Public Function IsEventLogFull(ByVal Source _
   As String, ByVal SourceMachine As String) _
   As Boolean
```

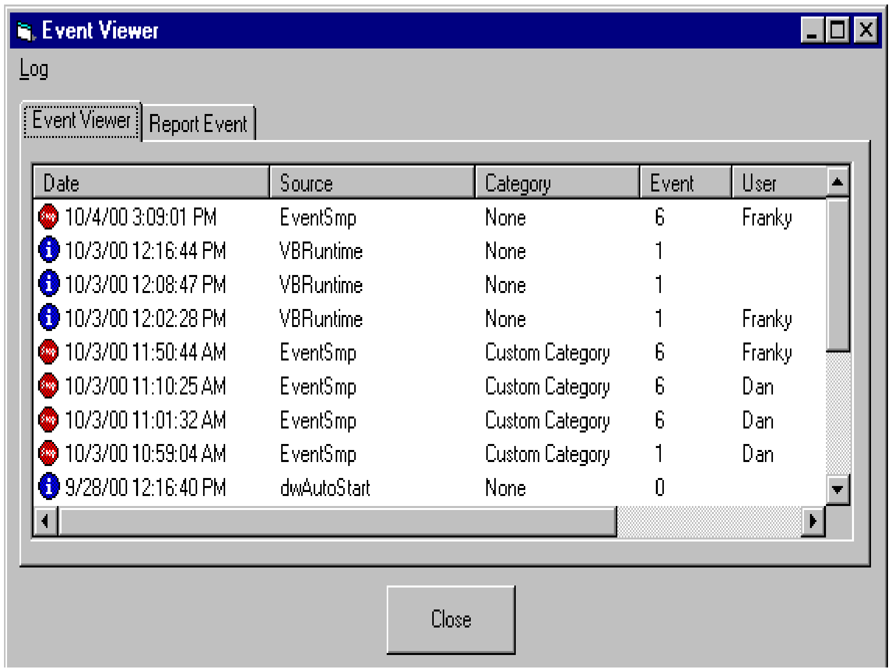
The parameters are as follows:

Parameter	Definition
Source	The name of the event source whose corresponding event log is to be tested.
SourceMachine	The name of the server containing the event source. Use an empty string for the current machine.

Using the Desaware Event Viewer and Reporter Utility

The Desaware Event Viewer project demonstrates how to use the Desaware Event Log class. It displays a list of the Events in the Event Viewer tab of the main form. You can double click on an entry to retrieve additional detailed information for each event – such as the event message and additional binary data. The Desaware Event Viewer displays similar information as the Microsoft Event Viewer included with Windows NT.

Note that displaying all the entries for a log file may take some time depending on how large the log file is.



Date	Source	Category	Event	User
10/4/00 3:09:01 PM	EventSmp	None	6	Franky
10/3/00 12:16:44 PM	VBRuntime	None	1	
10/3/00 12:08:47 PM	VBRuntime	None	1	
10/3/00 12:02:28 PM	VBRuntime	None	1	Franky
10/3/00 11:50:44 AM	EventSmp	Custom Category	6	Franky
10/3/00 11:10:25 AM	EventSmp	Custom Category	6	Dan
10/3/00 11:01:32 AM	EventSmp	Custom Category	6	Dan
10/3/00 10:59:04 AM	EventSmp	Custom Category	1	Dan
9/28/00 12:16:40 PM	dwAutoStart	None	0	

Figure 10
Desaware Event Viewer

Event Viewer

The icons displayed in each event entry identifies the severity of the events as follows:

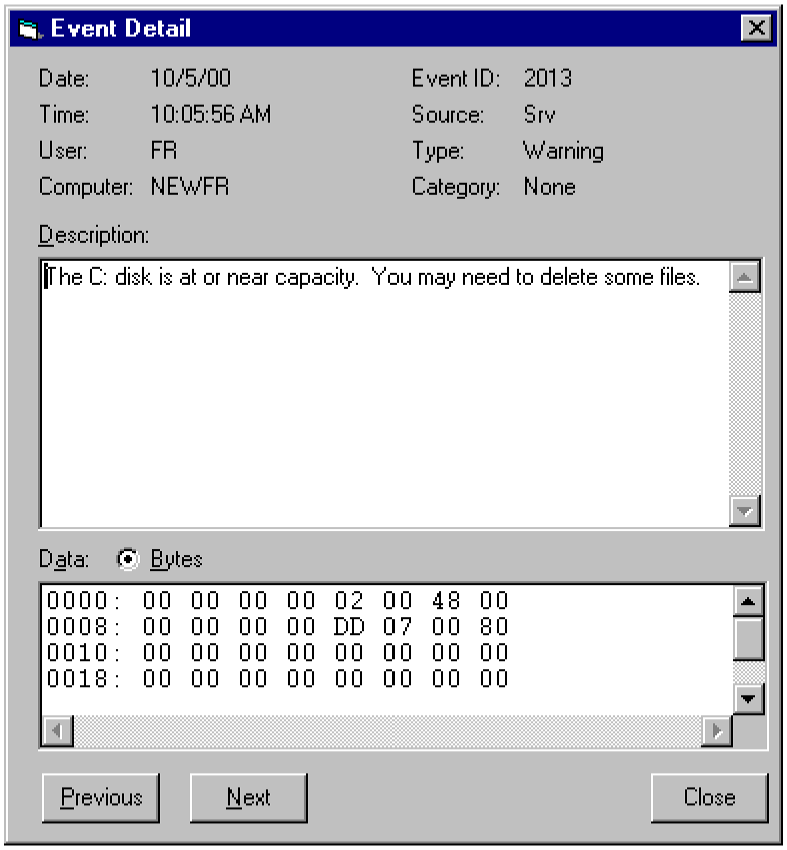
ℹ	Information
⛔	Error
⚠	Warning
🔒	Audit Failure
🔑	Audit Success

The following columns in the listview control display the following information:

Date	The date and time that the events were generated.
Source	The name of the Event Source for the events.
Category	The name of the Category for the events.

Event	The event number for the events.
User	The user name for the events, empty if no users were specified with the particular event.
Computer	The computer name for the events.

You can double click on a particular entry to view the message string or binary data for that event. Doing so will display the Event Detail form.


Event Detail

Date:	10/5/00	Event ID:	2013
Time:	10:05:56 AM	Source:	Srv
User:	FR	Type:	Warning
Computer:	NEWFR	Category:	None

Description:

The C: disk is at or near capacity. You may need to delete some files.

Data: (●) Bytes

```
0000:  00  00  00  00  02  00  48  00
0008:  00  00  00  00  DD  07  00  80
0010:  00  00  00  00  00  00  00  00
0018:  00  00  00  00  00  00  00  00
```

Figure 11
Desaware Event Log Event Detail Form

Event Detail Form

The Event Detail form displays the same information as presented in the listview control, with the addition of the message string and binary data for the selected event.

Select the **Next** button to view the detail event information for the next event record. Select the **Previous** button to view the detail event information for the previous event record. The string and binary data are presented as read-only. Select the **Close** button to exit this form.

Log Menu

The Log menu allows you to specify which type of event to display. You can choose among the Application, Security, or System events. You may also select to view the events for another computer. The **Select Computer** menu command displays a list of computers on your network from which you may select. The **Refresh** menu command causes the Event Viewer to read the event log file again and update the information displayed.

Report Event

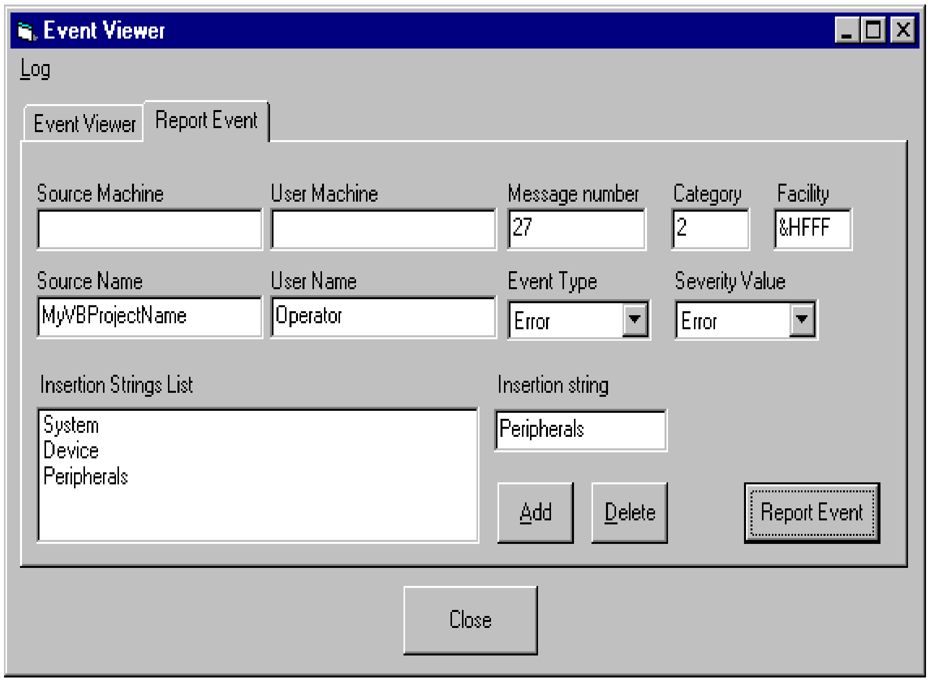
The Report Event tab allows you to report an event to the event log. The code demonstrates how to use the Desaware Event Log object's ReportEvent function.

Name	**Information**
Source Machine	Specify the name of the server containing the event source. Use an empty string for the current machine.
Source Name	Specify the name of the event source.
User Machine	Specify the server on which the user is valid.
User Name	Specify the name of the user to associate with the event.
Message Number	Specify the message number in the event source.
Category	Specify the category number in the event source, zero for no category.
Facility	Specify the facility value in the event source. To specify the facility in hexadecimal, precede the number with "&H".
Event Type	Select the severity of the event.
Severity Value	Select the severity value.

Insertion Strings List	Specify insertion strings for the event source. Insertion strings may be added by entering a string in the Insertion String text box, then selecting the **Add** button. Insertion strings values must begin with 1 and increment by 1 for each additional insertion string. You may delete an insertion string by selecting it and then selecting the **Delete** button.

The Facility, Message number, and Severity value are combined together to form the Event ID for the ReportEvent function.

The Report Event button calls the ReportEvent function to log the event.



Source Machine:
User Machine:
Message number: 27
Category: 2
Facility: &HFFF
Source Name: MyVBProjectName
User Name: Operator
Event Type: Error
Severity Value: Error
Insertion Strings List: System, Device, Peripherals
Insertion string: Peripherals

Figure 12
Desaware Event Viewer Form – Report Event Tab

Redistributable Components

Event sources that you create with this Toolkit are redistributable with no royalty fees.

You may incorporate the event log class into your applications and modify it as you wish, however you must include Desaware's copyright notice in the source code everywhere it appears. If you use this source code in a component that you wish to distribute or sell, you must add significant and primary functionality to the component (in other words – under this license you cannot market your own component whose primary task is reporting events into the event log using this source code).

No other files or components of this toolkit may be redistributed.

Technical Support

Desaware prides itself on providing excellent technical support at no charge.

At the same time, while we are glad to address any problems with our software, we know from experience that our software is often used in ways that we never imagined. As enabling technologies (i.e. technologies that allow VB programmers to do things that are beyond the typical VB application), we cannot characterize any of our components or tools for every possible application.

In other words, while we will do our best to address any bugs in our products or issues that look like they have the potential of being bugs, we cannot write your code for you, or debug your program for you. Nor can we provide one on one consulting on particular applications.

When you contact us, we will assume that you are familiar with the material in this manual. We ask that you reduce any problems to the smallest set of code that duplicates the problem.

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) 404-4760 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) 404-4760 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) 404-4760 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.

Index

Other Sources of Information

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

www.desaware.com

Desaware's web site includes numerous technical articles on all aspects of Windows development. Be sure to also check the FAQ and support section for this product.

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) 404-4760 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) 404-4760 or email support@desaware.com.

Moving to VB.Net: Strategies, Concepts and Code

Written by Daniel Appleman (president of Desaware) and published by Apress (ISBN# 1-893115-97-6).

VB.Net is not Visual Basic. COM+2.0 is not COM.

Porting is stupid. 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. He tackles strategic issues to help you determine when and whether to deploy VB.Net.

As always, Dan teaches the core concepts such as inheritance and multithreading, where VB6 programming habits can lead to costly design and development errors. And he covers the language changes to help you adapt to, and understand, this new environment.

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.

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 and web site (http://msdn.microsoft.com) contain 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.

Index

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 Standard 6/Professional 7.0

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 Professional Edition includes .NET support for keyboard hooks, window hooks and subclassing (including cross-task subclassing) with examples in both Visual Basic.NET and C#. Additionally, a WinSock component with comprehensive VB source code that gives you complete control for Internet/intranet programming.

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.

- 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 and ME depending upon which version (or edition) of SpyWorks.

STATECODER 1.0

A .NET class framework that makes it easy to create and support powerful state machines using VB .NET or C#. Dramatically improves the reliability of applications, components and services that make use of the multithreading and asynchronous features of .NET.

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!

- 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 2.0 COM Edition, .NET Edition

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 3.0

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 registration database controls for Windows NT, Windows 2000/XP, Windows 95 & 98. Plus, a simple resource compiler (with source) so that you can create your own .RES files for use with Visual Basic and more. 16 & 32 bit COM/ActiveX and .NET.

New for version 3.0! StorageTools 3.0 includes .NET support for accessing OLE Structure Storage from .NET assemblies.

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.