

LineGraph-5M

**Line and Scatter Plot Graphing Control
for ASP.NET**



Version 1

for Visual Studio .NET

Desaware, Inc.

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 to you for use only on a single computer. If you wish to install it on additional computers, you must purchase additional software licenses. You may (and should) make archival copies of the software for backup purposes.

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) 377-4770.

You have a royalty-free right to incorporate any of the sample code provided into your own applications with the stipulation that you agree that Desaware, Inc. has no warranty, obligation or liability, real or implied, for its performance.

Licensing: of the LineGraph-5M uses the Desaware Licensing System Component. This framework provides for the transfer of licensing information from the system upon which LineGraph-5M is installed, to Desaware's Licensing Web Service. This in turn creates and activates a license key that allows you to use the LineGraph-5M components on your system. The licensing information transferred is a one way cryptographic hash that does not include any personal information, or information that could be used to identify the originating system. If you perform online registration, the registration information will also be transferred.

File Descriptions: You may not distribute the following files with your compiled assemblies: LineGraph5m.dll, ReturnImage.aspx, ReturnImage.aspx.vb, WebForm1.aspx, WebForm1.aspx.vb, CustomControlTest.vbproj, ReturnImage.aspx, CustomControlTest.vbproj.webinfo, XMLFile1.xml, ReturnImage.aspx.vb, Web.config, Styles.css, Global.asax.vb, Global.asax, CustomControlTest.vsdisco, AssemblyInfo.vb, WebForm1.aspx.resx, Global.asax.resx. This software product is licensed on a 'per machine' basis.

Source Files: If you have purchased a source code license, the following applies: You may rebuild modified versions of the software provided subject to the restrictions listed. 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 the LineGraph-5M except for use with your own applications. Modified assemblies and namespaces must be renamed – you may not use Desaware in the assembly name or any namespace. However Desaware's copyright notice must be prominently displayed in any location where your own copyright notice is present. Source code may not be published or distributed, and may be used or accessed only by the individuals and at the locations specified by the source code license. You may distribute modified versions of the LineGraph5m.dll file only for use by your applications. Modified versions of other components may only be used by the individuals and at the locations covered by the source code license.

Limited Warranty

Desaware, Inc. warrants the physical CD 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 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 CD 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 Avenue, Suite 4, Campbell, CA 95008

Microsoft is a registered trademark of Microsoft Corporation. Visual Basic, Visual Studio, Windows, Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, and Windows XP are trademarks of Microsoft Corporation.

Desaware Licensing System, FiveMinuteSoftware, CAS/Tester, SpyWorks, NT Service Toolkit, StateCoder, VersionStamper, StorageTools, Event Log Toolkit, ActiveX Gallimaufry, Custom Control Factory, and SpyNotes #2, The Common Dialog Toolkit are trademarks of Desaware, Inc.

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.

Copyright © 2003 by Desaware, Inc. All rights reserved. Printed in the U.S.A.

Table of Contents

INTRODUCTION	6
Features	6
Installation	6
Samples	7
Licensing	7
USING LINEGRAPH-5M INTO YOUR ASPX ASSEMBLY	9
QUICK START	11
CONFIGURING THE LINEGRAPHSAMPLE PROJECT	14
Demo Installation	14
LINEGRAPH-5M REFERENCE	15
Enum PointStyles	15
Enum PointSizes	15
Enum GraphTypes	15
Enum GridTypes	16
Enum ImageFormats	16
Properties	16
Methods	27
PRODUCT AND BOOK CATALOG	34
<u>Software</u>	34
<u>Books by Dan Appleman</u>	37
<u>eBooks by Dan Appleman</u>	37

Introduction

The Desaware, Inc. LineGraph-5M component is designed to allow you to add line and scatter plot graphs to your ASP.NET pages. It can create graphs with an absolute minimum of effort, yet contains enough options to allow you to customize its appearance and function to suit varied needs. Numeric data can be plotted on both axis' and date/time information can be plotted on the X axis. All types of color and style attributes can be changed.

Features

- Easy to use - just supply collections of X and Y points or columns in a DataSet and LineGraph-M will draw a graph.
- Draw line graphs or scatter plot graphs.
- Customize graph colors, fonts, graph appearance, point styles, and many more optional properties.
- Able to use dates and times along the X axis with the display format of the values automatically adjusted according to the time span.
- 100% managed code.

As with all software in Desaware's Five Minute software line, the LineGraph control is easy to learn, use and deploy. Many developers will find the base component sufficient for most tasks, however a source code license is available for those who need additional customization.

Installation

Use the LineGraphSetup.msi or LineGraphSetup11.msi installation package to install the LineGraph-5M package on your system. The installation will display the following installation/registration screen:

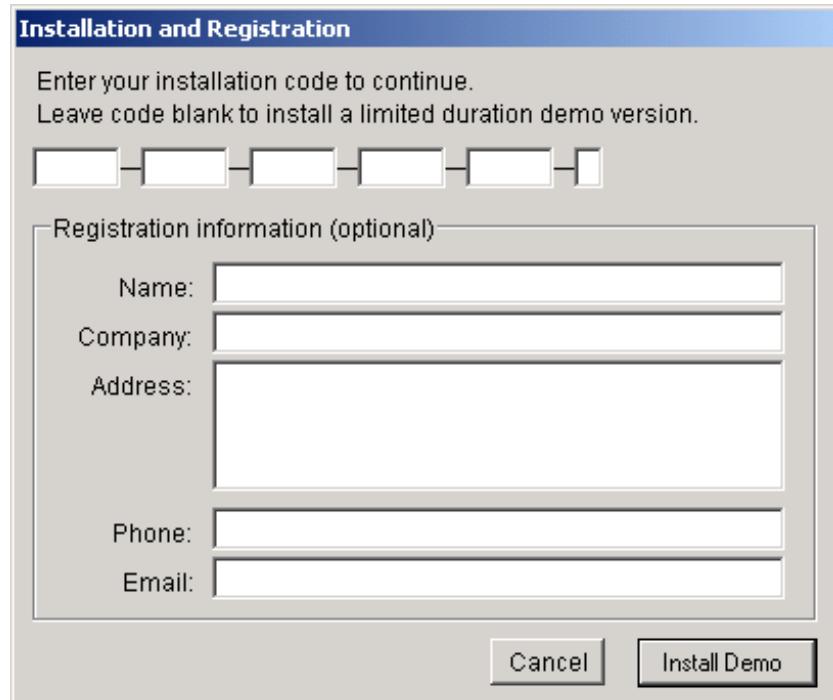


Figure 1
Install and Registration Dialog Box

If you do not enter a registration code, the application will be installed with a license valid for 30 days. If a demo version was previously installed and you wish to install a fully licensed copy, please uninstall and then reinstall the product using a valid installation code.

To register online, simply include the specified registration information. This will allow us to notify you about product updates and allow you to obtain free technical support.

Samples

The LineGraphSample web application demonstrates the majority of the features of the LineGraph control, and includes source code written in VB.NET and C#.

To install the sample application, click on the start menu and select the LineGraphSample installation link. Refer to the *Configuring the LineGraphSample* section of this document or the readme file for information regarding configuring this sample web application.

Licensing

The installation program will prompt you to enter your license key. If you do not provide a license key, you will be given a 30 day temporary license and the graphic "www.desaware.com" will appear in the background of the graph.

Once you install the product, the installer will create a license file (LineGraph5M10.dlsc) in the primary product directory.

You can use the LineGraph control by adding a reference to it from any ASP.NET project, or copying it to the executable directory for the application. However, you MUST also copy the LineGraph5M10.dlsc file for the system into that directory as well.

in addition to several dependent files - Desaware.MachineLicense.dll and Desaware.Dls.Interfaces.dll.

The LineGraph control is licensed for each machine. Once licensed on a system, you can use the control on as many ASP.NET applications as you wish on that system.

Using LineGraph-5M Into Your ASPX Assembly

To add graphing ability to your ASPX assembly, first add the LineGraph control to the ToolBox. Display the ToolBox, right click on the ToolBox window, and select “Customize Toolbox...” from the menu. This will display the “Customize Toolbox” dialog box. Select the “.NET Framework Components” tab, press “Browse” and locate the location of the Desaware.LineGraph.dll file. Press OK. You will now have an entry for LineGraph in the Web Forms section of the ToolBox (usually at the bottom).

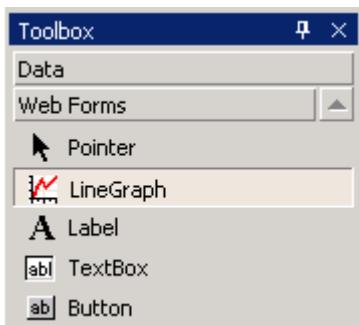


Figure 2
Toolbox Menu

You can select the LineGraph entry from the Toolbox menu and drop it on your ASPX page.

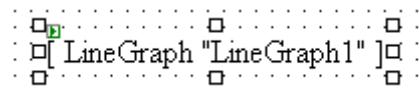


Figure 3
LineGraph Graphic

The LineGraph control generates images – it does not display them. To display an image, you must use the ReturnImage.aspx page to generate a link to the image. First ensure ReturnImage.aspx is part of your project. Then set the appropriate graph parameters. Call the CreateGraphSessionVariable() method, passing a unique name as the parameter. The control will then save the resulting image in a session variable with that name. To use the image, pass the name as the single parameter to the ReturnImage.aspx page, which will return a URL to the image.

[VB]

```
LineGraph5m1.CreateGraphSessionVariable("linegraph1")
' Image1 is a Image Web Control
Image1.ImageUrl = "ReturnImage.aspx?Name=linegraph1"
```

[C#]

```
LineGraph5m1.CreateGraphSessionVariable("linegraph1");
// Image1 is a Image Web Control
Image1.ImageUrl = "ReturnImage.aspx?Name=linegraph1";
```

Quick Start

If you have an array of X values and an array of Y values, you can create a graph.

Set the **ImageHeight** and **ImageWidth** properties (*see the following code example*), and call the **AddPointSet()** method with both arrays, and the component will create the graph. Call **CreateGraphFile()** to create a graph file or **CreateImage()** to draw a graph into a session variable which can be then accessed by ASPX pages.

[VB]

```
Imports Desaware.LineGraph  
' This line is generated for you when you place the control on  
' the aspx page.  
Dim LineGraph5M1 As New LineGraph()  
  
Dim x As New ArrayList()  
Dim y As New ArrayList()  
  
LineGraph5M1.ImageHeight = 120  
LineGraph5M1.ImageWidth = 220  
x.Add(10)  
x.Add(20)  
x.Add(35.6)  
y.Add(-5)  
y.Add(4.84)  
y.Add(7)  
LineGraph5M1.AddPointSet(x, y)  
LineGraph5M1.CreateGraphFile(Page.MapPath(".") + "\graph.gif")
```

[C#]

```
using Desaware.LineGraph;  
// This line is generated for you when you place the control on  
// the aspx page.  
LineGraph LineGraph5M1 = new LineGraph();  
  
ArrayList x = new ArrayList();  
ArrayList y = new ArrayList();  
  
LineGraph5M1.ImageHeight = 120;
```

```
LineGraph5M1.ImageWidth = 220;  
x.Add(10);  
x.Add(20);  
x.Add(35.6);  
y.Add(-5);  
y.Add(4.84);  
y.Add(7);  
LineGraph5M1.AddPointSet(x, y);  
LineGraph5M1.CreateGraphFile(Page.MapPath(".") + "\\graph.gif");
```

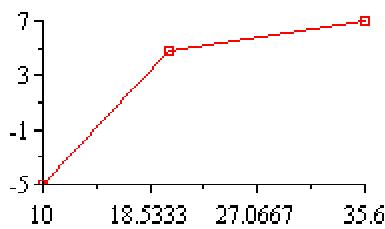


Figure 4
Results of Input (as a Graph)

To create a graph with no points, set the **ImageHeight** and **ImageWidth** properties, then set the **HigherXBound**, **LowerXBound**, **HigherYBound** and **LowerYBound** properties (*see the following code example*), and finally call the appropriate **CreateGraph()** / **CreateGraphFile()** / **CreateGraphBitmap()** method.

[VB]

```
LineGraph5m1.UseAutoSize = False  
LineGraph5m1.ImageHeight = 120  
LineGraph5m1.ImageWidth = 220  
LineGraph5m1.HigherXBound = 100  
LineGraph5m1.LowerXBound = 0  
LineGraph5m1.HigherYBound = 100  
LineGraph5m1.LowerYBound = 0  
LineGraph5m1.CreateGraphFile("graph.gif")
```

[C#]

```
LineGraph5m1.UseAutoSize = false;  
LineGraph5m1.ImageHeight = 120;  
LineGraph5m1.ImageWidth = 220;  
LineGraph5m1.HigherXBound = 100;  
LineGraph5m1.LowerXBound = 0;  
LineGraph5m1.HigherYBound = 100;  
LineGraph5m1.LowerYBound = 0;  
LineGraph5m1.CreateGraphFile("graph.gif");
```

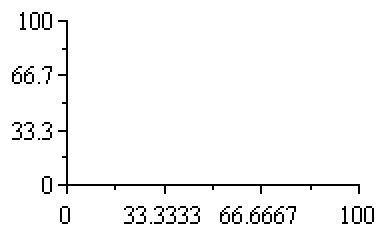


Figure 5
Results of Input (as a Graph)

Refer to the LineGraphSample web project for additional samples demonstrating the LineGraph control.

Configuring the LineGraphSample Project

The LineGraphSample project demonstrates the majority of the features of the LineGraph control.

We gave some thought into whether to completely automate the installation, but decided it would be a better experience to actually require you to build and explicitly configure the web application.

If you wish to automate the installation/configuration of web application databases that you deploy, check out our InstallationHelper-5M product at <http://www.5MinuteSoftware.com>.

Demo Installation

To install and configure the LineGraphSample project, do the following:

1. Be sure your ASP.Net compatible web server is running.
2. Using the Program/Start menu, install the LineGraphSample.msi installer file.
3. After installation is completed, copy the LineGraph license file (LineGraph5M10.dlsc) from your LineGraph “bin” folder to the location of the bin folder of the new web application.
4. Open the project file. Typically <http://localhost/LineGraphSample/LineGraphSample.vbproj>.
5. You should now be able to run the application. Set the default page if prompted to WebForm1.aspx.

LineGraph-5M Reference

The following is detailed reference information for the Desaware LineGraph class. Properties and methods of base classes are not listed unless they are overridden.

Note on thread safety: Unless otherwise noted, as is common in the .NET Framework, static methods of objects are thread safe. Instance methods are not.

Enum PointStyles

A list of symbols to be drawn at each point. The possible values are:

Symbol	Value
Empty	null value, no point drawn
None	no point drawn
SinglePixel	single pixel is drawn
Plus	plus sign
Circle	hollow circle
Diamond	hollow diamond
Square	hollow square
Pyramid	hollow triangle with point up
Triangle	hollow triangle with point down
CircleF	filled circle
DiamondF	filled diamond
SquareF	filled square
PyramidF	filled triangle with point up
TriangleF	filled triangle with point down

Enum PointSizes

Size of **PointStyles** symbols (except *SinglePixel*, which is always a single pixel).

- Small
- Medium
- Large

Enum GraphTypes

This is the type of graph to draw. *ScatterPlot* only draws the point symbols, while *Line* draws point symbols and connecting lines.

- ScatterPlot
- Line

Enum GridTypes

This is the type of grid drawn on the graph. Grid lines are light gray and are drawn from the location of each major tick.

- None
- Vertical
- Horizontal
- Both

Enum ImageFormats

The file format of the graph, used in both **CreateGraph()** and **CreateGraphFile()** methods. The *PNG* format usually results in a better image, but not all browsers support that format. *GIF* is better supported, but the drawing routines will often be dithered unless only the most basic colors are used. *JPEG* format is not supported by the LineGraph control as .NET support for that format results in unacceptably blurry lines and text.

- GIF
- PNG

Properties

ImageWidth	<pre>[VB] Property ImageWidth As Integer [C#] int ImageWidth</pre>
	<p>Represents the size, in pixels, of the image created by the component. This must be set before a graph can be drawn. If the number is too small, an unsatisfactory image may result. Values below 50 will generate an exception.</p>
ImageHeight	<pre>[VB] Property ImageHeight As Integer [C#] int ImageHeight</pre>
	<p>Represents the size, in pixels, of the image created by the component. This must be set before a graph can be drawn. If the number is too small, an unsatisfactory image may result. Values below 50 will generate an exception.</p>
ImageFormat	<pre>[VB] Property ImageFormat As ImageFormats [C#] ImageFormats ImageFormat</pre>
	<p>Instance of the ImageFormats enumeration. Describes the file format of the image created by the component. This impacts both CreateGraph() and CreateGraphFile() methods. See the ImageFormats enumeration for detailed information. PNG</p>

format supports more colors, but is not supported by all web browsers. GIF files are supported by all browsers, but colors can be dithered (that is, comprised of pixels of different basic colors) and may look muddy. To prevent dithering in GIF images, use basic colors (Red or Green instead of Salmon or Forest). By default images are in GIF format.

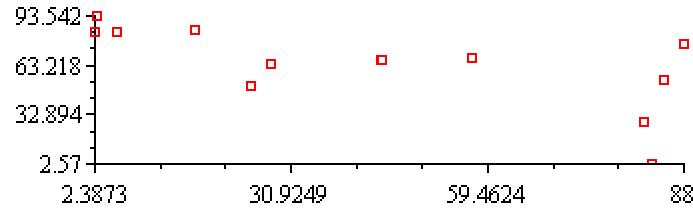
```
LineGraph5M1.ImageFormat =  
LineGraph.ImageFormats.PNG
```

GraphType	[VB] Property GraphType As GraphTypes [C#] GraphTypes GraphType
-----------	--------------------------------------------------------------------

Instance of the **GraphTypes** enumeration. Describes the type of graph being drawn, either line graph (lines connect each point symbol) or scatter plot graph (where only point symbols are drawn). The default value is LineGraph. See the **GraphTypes** enumeration for more information.

To draw lines without point symbols, set the Point parameter to None when you call the **AddPointSet** method, or use the **Sets** property to change the symbol to None after the set has been added.

```
LineGraph5M1.GraphType =  
LineGraph.GraphTypes.ScatterPlot
```



HigherXBound	[VB] Property HigherXBound As Object [C#] object HigherXBound
--------------	------------------------------------------------------------------

Only numeric types (integers and floating point) and the DateTime type are accepted. This is the highest boundary of the X axis of the graph. If the **UseAutoSize** property is set to True, then this value can only be read and not modified. If sets of points are added after this property is set, the property may be adjusted automatically to fit the new points.

HigherYBound	[VB] Property HigherYBound As Double [C#] double HigherYBound																								
	<p>This is the higher bound of the Y axis of the graph. If the UseAutoSize property is set to True, then this value can only be read and not modified.</p>																								
LowerXBound	[VB] Property LowerXBound As Object [C#] object LowerXBound																								
	<p>Only numeric types (integers and floating point) and the DateTime type are accepted. This is the lower boundary of the X axis of the graph. If the UseAutoSize property is set to True, then this value can only be read and not modified. If sets of points are added after this property is set, the property may be adjusted automatically to fit the new points.</p>																								
LowerYBound	[VB] Property LowerYBound As Double [C#] double LowerYBound																								
	<p>This is the lower bound of the Y axis of the graph. If the UseAutoSize property is set to True, then this value can only be read, and attempts to change it will not have an effect. If sets of points are added after this property is set, the property may be adjusted automatically to fit the new points.</p>																								
	<pre>LineGraph5M1.LowerXBound = 0 LineGraph5M1.HigherXBound = 120 LineGraph5M1.LowerYBound = 0 LineGraph5M1.HigherYBound = 120</pre>																								
	<table border="1"> <caption>Data points estimated from the graph</caption> <thead> <tr> <th>X</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>85</td></tr> <tr><td>10</td><td>85</td></tr> <tr><td>20</td><td>85</td></tr> <tr><td>30</td><td>45</td></tr> <tr><td>40</td><td>75</td></tr> <tr><td>50</td><td>75</td></tr> <tr><td>60</td><td>75</td></tr> <tr><td>70</td><td>75</td></tr> <tr><td>80</td><td>35</td></tr> <tr><td>85</td><td>55</td></tr> <tr><td>90</td><td>85</td></tr> </tbody> </table>	X	Y	0	85	10	85	20	85	30	45	40	75	50	75	60	75	70	75	80	35	85	55	90	85
X	Y																								
0	85																								
10	85																								
20	85																								
30	45																								
40	75																								
50	75																								
60	75																								
70	75																								
80	35																								
85	55																								
90	85																								

MajorTicksXAxis	<pre>[VB] Property MajorTicksXAxis As Integer [C#] int MajorTicksXAxis</pre>
	<p>This is the number of large ticks on the X axis of the graph. The large ticks are labeled, if there is room for the label. Grids are drawn from each major tick. This value is always at least 2. It is set to 4 by default.</p>
MajorTicksYAxis	<pre>[VB] Property MajorTicksYAxis As Integer [C#] int MajorTicksYAxis</pre>
	<p>This is the number of large ticks on the Y axis of the graph. The large ticks are labeled, if there is room for the label. Grids are drawn from each major tick. This value is always at least 2. It is set to 4 by default.</p>
MinorTicksXAxis	<pre>[VB] Property MinorTicksXAxis As Integer [C#] int MinorTicksXAxis</pre>
	<p>This is the number of small ticks <i>between</i> each major tick on the X axis of the graph. Has a value of 1 by default. Setting this to zero would remove minor ticks from the X axis.</p>
MinorTicksYAxis	<pre>[VB] Property MinorTicksYAxis As Integer [C#] int MinorTicksYAxis</pre>
	<p>This is the number of small ticks <i>between</i> each major tick on the Y axis of the graph. Has a value of 1 by default. Setting this to zero would remove minor ticks from the Y axis.</p>
<pre>LineGraph5M1.MajorTicksXAxis = 5 LineGraph5M1.MajorTicksYAxis = 5 LineGraph5M1.MinorTicksXAxis = 4 LineGraph5M1.MinorTicksYAxis = 4</pre>	

Sets

```
[VB] Dim Sets As SetType()
```

```
[C#] SetType[] Sets
```

Array of structures describing each set of points. Properties of the structure are:

LineColor As Color - color of lines in line graphs.

PointColor As Color - color of points when points are drawn.

PointStyle As PointStyles - type of point to draw.

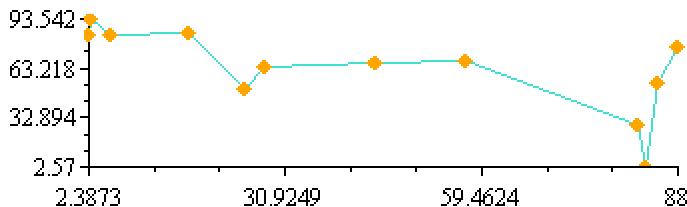
Once a set of points is added to the graph, you can use this array to modify its properties.

```
[VB]
```

```
LineGraph5M1.Sets(0).LineColor = Color.Turquoise  
LineGraph5M1.Sets(0).PointColor = Color.Orange  
LineGraph5M1.Sets(0).PointStyle = LineGraph.  
PointStyles.DiamondF
```

```
[C#]
```

```
LineGraph5M1.Sets[0].LineColor = Color.Turquoise;  
LineGraph5M1.Sets[0].PointColor = Color.Orange;  
LineGraph5M1.Sets[0].PointStyle = LineGraph.  
PointStyles.DiamondF;
```

**NumSets**

```
[VB] ReadOnly Property NumSets As Integer
```

```
[C#] int NumSets [get]
```

Returns the number of line sets the component currently holds.

```
[VB]
```

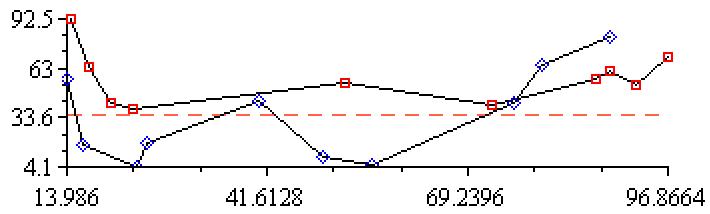
```
' Make all lines on the graph black.
```

```

Dim i As Integer
For i = 0 to LineGraph5M1.NumSets - 1
    LineGraph5M1.Sets(i).LineColor = Color.Black
Next I

[C#]
// Make all lines on the graph black.
int i;
for (i = 0; i < LineGraph5M1.NumSets; i++)
    LineGraph5M1.Sets[i].LineColor = Color.Black;

```

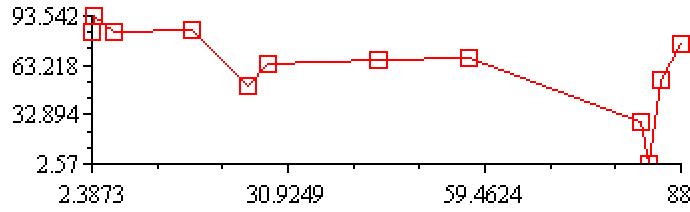


PointSize

[VB] Property PointSize As PointSizes
[C#] PointSizes PointSize

Expresses the size of the point symbols drawn (except for the "single point" style). See the PointSizes enumeration for more information. By default the size is "small".

LineGraph5M1.PointSize = LineGraph.PointSizes.Large

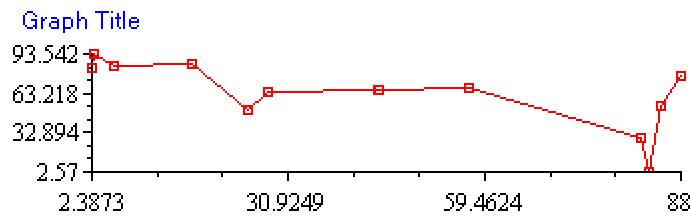


TitleCaption

[VB] Property TitleCaption As String
[C#] string TitleCaption

Text that is displayed on the top left corner of the graph. Its color is set by the **CaptionColor** property and its font is set by the **Font** property. By default no title is shown, and the graph is expanded to fill the space.

```
LineGraph5M1.TitleCaption = "Graph Title"
```

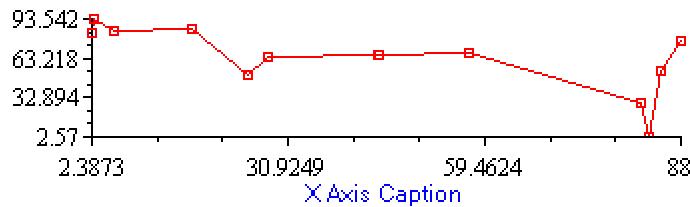


XAxisCaption

```
[VB] Property XAxisCaption As String  
[C#] string XAxisCaption
```

Text that is displayed along the X axis (bottom of the graph). Its color is set by the **CaptionColor** property and its font is set by the **Font** property. By default no caption is shown, and the graph is expanded to fill the space.

```
LineGraph5M1.XAxisCaption = "X Axis Caption"
```

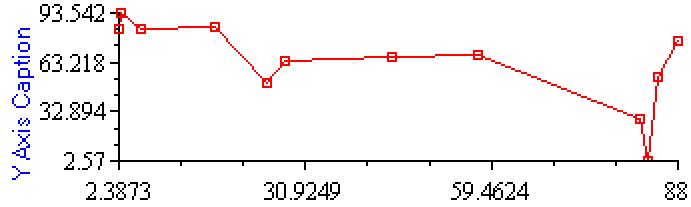


YAxisCaption

```
[VB] Property YAxisCaption As String  
[C#] string YAxisCaption
```

Text that is displayed along the Y axis (left of the graph). It is displayed vertically. Its color is set by the **CaptionColor** property and its font is set by the **Font** property. By default no caption is shown, and the graph is expanded to fill the space.

```
LineGraph5M1.YAxisCaption = "Y Axis Caption"
```



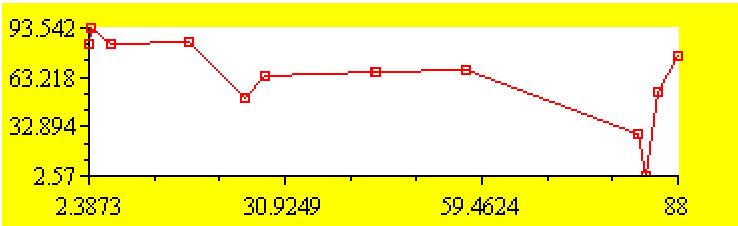
BackColor

[VB] Property BackColor As Color

[C#] Color BackColor

Background color of the image. By default it is set to Color.White (255,255,255 RGB).

```
LineGraph5M1.BackColor = Color.Yellow
```



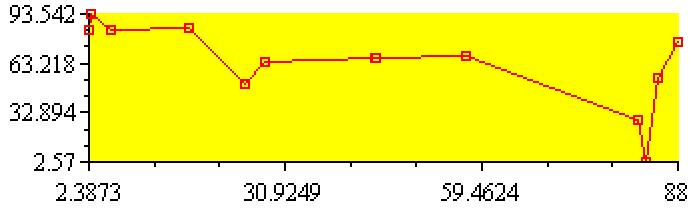
ForeColor

[VB] Property ForeColor As Color

[C#] Color ForeColor

Color of the area contained within the graph bounds. By default it is set to Color.White (255,255,255 RGB).

```
LineGraph5M1.ForeColor = Color.Yellow
```



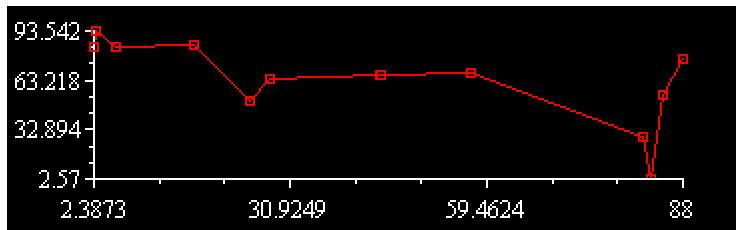
AxisColor

[VB] Property AxisColor As Color

[C#] Color AxisColor

Color of the axis and the text at the tick marks. By default it is Color.Black (0,0,0 RGB).

```
LineGraph5M1.BackColor = Color.Black  
LineGraph5M1.GraphBackColor = Color.Black  
LineGraph5M1.AxisColor = Color.White
```



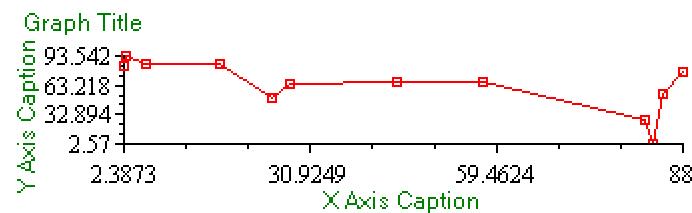
CaptionColor

[VB] Property CaptionColor As Color

[C#] Color CaptionColor

Color of **XAxisCaption**, **YAxisCaption**, and **TitleCaption**. By default it is set to Color.Blue (0,0,255 RGB).

```
LineGraph5M1.CaptionColor = Color.Green
```



IndicatorColor

[VB] Property IndicatorColor As Color

[C#] Color IndicatorColor

Color of the dashed lines added by **AddIndicationLine()** and areas added by **AddIndicationArea()** methods. By default this property is set to Color.Tomato (RGB 255, 99, 71).

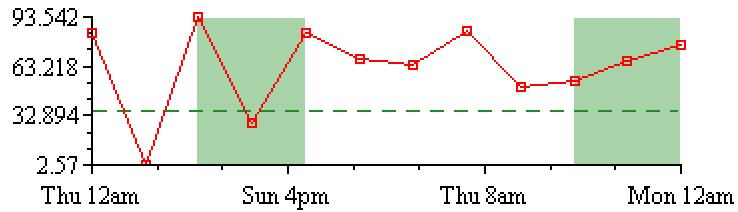
[VB]

```

LineGraph5M1.AddIndicatorLine(35.5)
    ' Mark the weekends
    LineGraph5M1.AddIndicatorArea(New
        DateTime(2003,5,3), New DateTime(2003,5,5))
    LineGraph5M1.AddIndicatorArea(New
        DateTime(2003,5,10), New DateTime(2003,5,12))
    LineGraph5M1.IndicatorColor = Color.ForestGreen

[C#]
LineGraph1.AddIndicatorLine(35.5);
// Mark the weekends
LineGraph1.AddIndicatorArea(new DateTime(2003,5,3),
    new DateTime(2003,5,5));
LineGraph1.AddIndicatorArea(new
    DateTime(2003,5,10), new DateTime(2003,5,12));
LineGraph1.IndicatorColor = Color.ForestGreen;

```



UseGrid

[VB] Property UseGrid As GridTypes

[C#] GridTypes UseGrid

Describes the type of grid lines drawn on the graph. The grid is a series of light grey lines drawn along the major tick marks, either vertically, horizontally, or both. See the Grid enumeration for detailed information. By default, no grid is drawn.

LineGraph5M1.UseGrid = LineGraph.GridTypes.Both

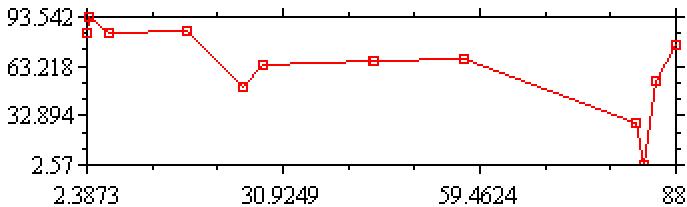


UseBoxAxis

[VB] Property UseBoxAxis As Boolean
 [C#] bool UseBoxAxis

If set to True, the axis and tick marks are drawn on all sides.
 Set to False by default.

```
LineGraph5M1.UseBoxAxis = True
```



UseProportionalFont

[VB] Property UseProportionalFont As Boolean
 [C#] bool UseProportionalFont

If set to True, the font size of XAxisCaption, YAxisCaption, and TitleCaption will change depending on the size of the graph image. Set to False by default.

UseAutoSize

[VB] Property UseAutoSize As Boolean
 [C#] bool UseAutoSize

If set to True, then the graph limits are set to perfectly match the points of all the sets and the LowerXBound, LowerYBound, HigherXBound, HigherYBound properties are ignored. If set to False, then LowerXBound, LowerYBound, HigherXBound, HigherYBound must be set. The higher bounds must be larger than the lower bounds or the graph will not be drawn. Set to True by default.

UsesDateXAxis

[VB] ReadOnly Property UsesDateXAxis As Boolean

[C#] bool UsesDateXAxis [get]

Read only Boolean. If set to True, then the graph is using DateTime types on the X axis instead of numeric values. The graph determines this by checking the type of the first point given it - from that point onwards, it ignores data that does not correspond to that type.

Methods

AddPointSet

```
[VB] Sub AddPointSet  
    (ByVal X As ICollection, ByVal Y As ICollection)  
    (ByVal X As ICollection, ByVal Y As ICollection, ByVal  
     SetColor As Color)  
    (ByVal X As ICollection, ByVal Y As ICollection, ByVal  
     SetColor As Color, ByVal PointStyle As PointStyles)  
    (ByVal X As ICollection, ByVal Y As ICollection, ByVal  
     LineColor As Color, ByVal PointColor As Color)  
    (ByVal X As ICollection, ByVal Y As ICollection, ByVal  
     LineColor As Color, ByVal PointColor As Color, ByVal  
     PointStyle As PointStyles)  
  
[C#] void AddPointSet  
    (ICollection X, ICollection Y)  
    (ICollection X, ICollection Y, Color SetColor)  
    (ICollection X, ICollection Y, Color SetColor,  
     PointStyles PointStyle)  
    (ICollection X, ICollection Y, Color LineColor, Color  
     PointColor)  
    (ICollection X, ICollection Y, Color LineColor, Color  
     PointColor, PointStyles PointStyle)
```

Adds a set of points to the graph. The *X* and *Y* parameters can be of any type that supports the ICollection interface, such as Collections, Arrays, ArrayLists and many other types.

The *X* collection must be a collection of numeric types (integer or floating point numbers) or DateTime items. The *Y* collection must be a collection of numeric types. Any item in either collection that does not correspond to this limitation will not be added to the points on the graph. The first item in the first point set added determines the type of the X axis (either a number or a date) – any x value that does not correspond to the selected type will also cause the pair to

	<p>be ignored.</p> <p>Overrides allow you to specify the color and point styles of the points and/or lines. <i>SetColor</i> is the color of both the line and the point symbols. If no color or point style is specified, unique ones will automatically be set.</p> <pre>[VB] Dim x As New ArrayList() Dim y As New ArrayList() x.Add(10) x.Add(20) x.Add(35.6) y.Add(-5) y.Add(4.84) y.Add(7) LineGraph5M1.AddPointSet(x, y, Color.Green, _ LineGraph.LineStyle.None) [C#] ArrayList x = new ArrayList(); ArrayList y = new ArrayList(); x.Add(10); x.Add(20); x.Add(35.6); y.Add(-5); y.Add(4.84); y.Add(7); LineGraph5M1.AddPointSet(x, y, Color.Green, _ LineGraph.LineStyle.None);</pre>
DataBind	<pre>[VB] Sub DataBind() [C#] void DataBind()</pre> <p>The LineGraph5m component can accept data from DataSets as well as collections. To do this, first the DataSource property is set</p>

to the DataSet where the data will originate. The **DataMember** property is set to the text name of the particular Table in which the data resides. The **DataFieldX** and **DataFieldY** properties are then set to the string name of the columns where the X and Y positions of the points are located. Once the properties are correctly set, call the **.DataBind()** method to add the point set. You cannot change the attributes of a set when it is added - after it is loaded you can use the **Sets** property as needed.

XML file is formatted as follows:

```
<XMLSchema1 xmlns="http://tempuri.org/XMLSchema1.xsd">
<Position xmlns="http://tempuri.org/XMLSchema1.xsd">
    <X>5.63</X>
    <Y>84.27</Y>
    <Date>2003-05-01T00:00:00.000000-07:00</Date>
</Position>
```

and so on

[VB]

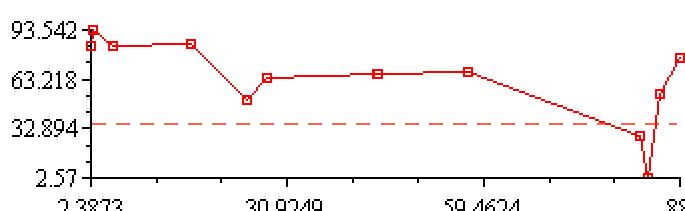
```
Dim ds as New DataSet()
ds.ReadXml("XMLFile1.xml")
LineGraph5M1.DataSource = ds
LineGraph5M1.DataMember = "Position"
If (UsingDates = True) Then
    LineGraph5M1.DataFieldX = "X"
Else
    LineGraph5M1.DataFieldX = "Date"
End If
LineGraph5M1.DataFieldY = "Y"
LineGraph5M1.DataBind()
' Don't draw point symbols with this line.
LineGraph5M1.Sets(LineGraph5M1.NumSets - 1).PointStyle
= LineGraph.PointStyles.None
```

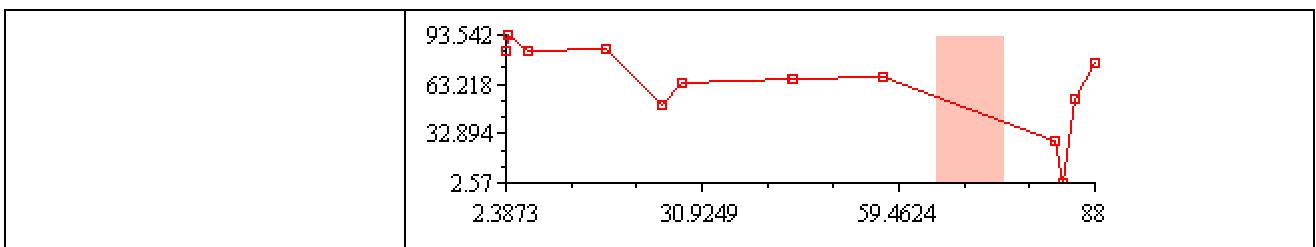
[C#]

```
DataSet ds = new DataSet();
ds.ReadXml("XMLFile1.xml");
LineGraph5M1.DataSource = ds;
```

	<pre> LineGraph5M1.DataMember = "Position"; if (UsingDates) LineGraph5M1.DataFieldX = "X"; else LineGraph5M1.DataFieldX = "Date"; LineGraph5M1.DataFieldY = "Y"; LineGraph5M1.DataBind(); // Don't draw point symbols with this line. LineGraph5M1.Sets[LineGraph5M1.NumSets - 1].PointStyle = LineGraph.PointStyles.None; </pre>
CreateGraphSessionVariable	<p>[VB] Sub CreateGraphSessionVariable (SessionVariableName As String)</p> <p>[C#] void CreateGraphSessionVariable (string SessionVariableName)</p> <p>Creates the graph image file and saves it to a memory stream. This is useful for creating an .aspx file that returns a link to an image. The image is in the file format specified in the ImageFormat property. If there is an error, no image will be written. For a graph to be drawn, you must do at least one of the following (besides indicating the size of the image):</p> <ul style="list-style-type: none"> • Set the HigherXBound, LowerXBound, HigherYBound, LowerYBound properties • Add at least 2 points to the graph. <p>If the high boundaries are equal to or smaller than the low boundaries, the image size is too small, or there was an error writing the file, an exception is generated.</p> <p>The sample code below calls a simple ASPX page that only loads the image from the specified session variable and passes back the URL to it so that the image control can show it.</p> <pre> LineGraph5M1.CreateGraphSessionVariable("DataSetGraph") Image1.ImageUrl = "ReturnImage.aspx?Name=DataSetGraph" </pre>

CreateGraphFile	<pre>[VB] Sub CreateGraphFile (ByVal filename As String) [C#] void CreateGraphFile (string filename)</pre> <p>Creates the graph image file and saves it to the path and filename specified. This can be useful in cases where you don't need to generate a custom graph each time a page is viewed. If the name does not have the correct extension, one will be created matching the file format. If there is an error, no image will be written.</p> <p>For a graph to be drawn, you must do at least one of the following (besides indicating the size of the image):</p> <ul style="list-style-type: none"> • Set the HigherXBound, LowerXBound, HigherYBound, LowerYBound properties • Add at least 2 points to the graph. <p>If the high bounds are equal to or smaller than the low bounds, the image size is too small, or there was an error writing the file, an exception is generated.</p> <pre>LineGraph5M1.CreateGraphFile(Page.MapPath(".") + "\graph.gif")</pre>
CreateGraphBitmap	<pre>[VB] Function CreateGraphBitmap () As Bitmap [C#] Bitmap CreateGraphBitmap ()</pre> <p>Creates the graph image file and returns a .NET Framework Bitmap object. This can be useful if you want to add your own drawings to the graph, or want to use a component that takes a Bitmap or some sort of array or block of data (which a Bitmap can easily be converted to using the Bitmap.Save() method).</p> <p>For a graph to be drawn, you must do at least one of the following (besides indicating the size of the image):</p> <ul style="list-style-type: none"> • Set the HigherXBound, LowerXBound, HigherYBound, LowerYBound properties • Add at least 2 points to the graph. <p>If the high bounds are equal to or smaller than the low bounds, the image size is too small, or there was an error writing the file, an exception is generated.</p>

	<p>Be sure you include "System.Drawing" in order to use the Bitmap object.</p> <pre>bitmapobject = LineGraph5M1.CreateGraphBitmap()</pre>
AddIndicatorLine	<p>[VB] Sub AddIndicatorLine (ByVal ValueY As Double) [C#] void AddIndicatorLine (double ValueY)</p> <p>Adds a dashed line at the particular Y value across the width of the graph. This is useful for indicating upper and lower limits of point sets, averages, a zero value, or a previous day's stock value to compare to the current graph. You can chance the color of the Indicator Line by using the IndicatorColor property.</p> <pre>LineGraph5M1.AddIndicatorLine (35)</pre> 
AddIndicatorArea	<p>[VB] Sub AddIndicatorArea (ByVal Start As Double, ByVal Finish As Double) (ByVal Start As DateTime, ByVal Finish As DateTime)</p> <p>[C#] void AddIndicatorArea (double start, double finish) (DateTime start, DateTime finish)</p> <p>Adds a shaded area between the specified X values along the height of the graph. Useful for indicating areas of the graph that are special or speculative. You can chance the color of the Indicator Line by using the IndicatorColor property.</p> <pre>LineGraph5M1.AddIndicatorArea (65, 75)</pre>



Product and Book Catalog

We would like to invite you to visit our Web site at: www.desaware.com for detailed product and book descriptions, product demos, FAQ pages and additional technical articles.

Software

Desaware Licensing System™

The Desaware Licensing System is a cryptographic based licensing system for .NET.

Designed for per server/machine and component licensing, it is extremely easy to use and can be configured for both moderate and high security scenarios. With 128 bit end to end cryptographic licensing, the Desaware Licensing System does not depend on hidden files, registry entries or other invasive techniques.

CAS/TesteTM

CAS/TesteTM is an automated Code Access Security tester for .NET assemblies. Because you can never be certain what permissions are allowed on a target system, it is essential that you test your assemblies under a variety of configurations and make sure it will fail gracefully regardless of how a system is configured. CAS/TesteTM executes your assembly on multiple configurations (over 80 predefined, and no limit to the number you can define). Detailed reports show exactly what exceptions occur under each scenario and where.

For components, class libraries, Windows Forms controls, Windows Forms applications and console applications – CAS/TesteTM will help your developers implement code access security, and testers verify behavior.

StateCoderTM

StateCoder is a .NET namespace that is designed to make it easy to create and support powerful State Machines in .NET using Visual Basic .NET, C# and other .NET Languages.

With Desaware's StateCoder, you will create .NET code that is more reliable, easier and cheaper to test, support, understand and modify.

With sophisticate thread management, it is ideal for creating multithreaded applications and component, including asynchronous design patterns, background operations, and protocols.

SpyWorks Professional and Standard Editions

You *can* do that in Visual Basic—*SpyWorks* is the tool that lets you do things in VB, VB.Net and C# that are normally not possible.

- Use hooks to detect messages or keystrokes for selected windows or the entire system.
- Export functions from Visual Basic DLL's, **C# orVB.NET DLL assemblies!**
- Use advanced subclassing techniques including the ability to subclass other applications.
- COM edition supports a wide variety of sophisticated operations, allowing you to do virtually anything in VB6 that is possible using VC++.

And much more—Visit our Web site for details.

NT Services Toolkit

Available in both COM and .NET editions.

The .NET edition provides numerous features beyond those provided by the service framework included with .NET. Among these are the ability to easily expose objects simultaneously via DCOM and .NET Remoting, the ability to test the service without actually installing it as a service, and self-installation features.

The COM edition allows you to create powerful services with VB6 – you can even run services and debug them from within the VB6 environment.

Desaware Event Log Toolkit

Desaware's *Event Log Toolkit* makes creation of custom event sources easy, and provides all the tools needed to create and log custom events. Both VB6 and .NET developers will benefit from the ability to precisely define event log messages, severity and categories. Includes VB6 classes (with source) for advanced event log management and reporting.

VersionStamper

Safely distribute your COM component-based applications, avoiding “DLL-Hell”. VersionStamper verifies that applications have the correct versions of DLL, OCX and other components, offering remote diagnostics, reporting and the ability to make your applications and components self-updating.

VersionStamper has saved our customers fortunes in support costs. Download our demo and find out for yourself how they did it.

StorageTools

OLE Structured Storage is the technology Microsoft uses with their own office applications to store multiple streams of complex data into a single file. Easier to use than a database, and more flexible for hierarchical data storage, StorageTools is the key to using structured storage from within Visual Basic, C#, VB .NET and other .NET languages.

The Desaware ActiveX Gallimaufry

This eclectic set of ActiveX controls written in Visual Basic is both useful and educational. Includes full VB source code. These include: a taskbar control, common dialog controls, TWAIN control (for scanners and digital cameras) and more. Includes full VB6 source.

5 Minute Software™ by Desaware

Desaware's new line of software is designed so you can learn, build and deploy solutions in about five minutes. Providing simple, targeted solutions to common problems, Desaware's 5-Minute Software line is a new approach to .NET components.

OneTimeDownload-5M™

OneTimeDownload-5M is a component that allocates, implements and manages temporary links. These are URL's that are active for a limited amount of time, and are often uniquely associated with individual users. Typical scenarios for this type of link include implementing special offers or discounts, software or document distribution, email campaign tracking and more.

IniFileTool-5M™

While XML is often used for application configuration, it can be notoriously difficult for end-users to edit. A single minor error can prevent an entire XML file from loading.

INIFileTool-5M makes it easy for you to read and write INI files from your .NET applications or web applications. Not only does it avoid the need for API calls, but more important, it is a 100% managed code solution that does not use API calls, thus is able to run in partial trust scenarios

InstallationHelper-5M™

The Visual Studio Deployment project offers an easy to use method for installing ASP.Net web applications and services. However, there are a number of common tasks that are important, but difficult to accomplish. From creating and configuring databases during installation (SQL or Access), to configuring IIS, to writing custom configuration data, InstallationHelper-5M is the missing piece for many installation tasks.

Books by Dan Appleman

Moving to VB.NET: Strategies, Concepts and Code, 2nd ed.

Written for Visual Basic 6 developers who are ready to learn and migrate to .NET, this book is ideal to not only get you started, but give you the foundation you need to progress further.

In Strategies you'll learn when and why to migrate to .NET, and when you shouldn't. In Concepts you'll not only learn new concepts, but will unlearn old design patterns that can get you into trouble. And in Code you'll learn about the changes to the language itself, along with a thorough introduction to the .NET framework, including key concepts such as distributed programming and security.

How Computer Programming Works

This fully illustrated beginner's book is the perfect book for friends, family and kids – anyone who would like to know more about programming. It's the book to read before you get a programming book, teaching key concepts like variables, compilers, program flow, etc. Think of it as a computer science course for everyone.

eBooks by Dan Appleman

Hijacking .NET

Each ebook in this series explores ways you can make use of undocumented or hidden capabilities within the .NET framework.

- Volume 1 discusses role based security, showing how to use private functions to enumerate roles for an account and set security for files and directories.

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

In this best-selling E-book, you will find an in-depth comparison of the two languages. In a feature by feature, head to head contest, you'll learn there really is a best choice, but that it depends on your specific situation.

Obfuscating .NET: Protecting your code from prying eyes

Did you know that you ship your complete source code any time you distribute a .NET assembly?

One of the consequences of the architecture of .NET is that a great deal of information about an assembly is kept with the assembly in a part of the file called the Manifest. This information makes it remarkably easy to not just recompile the assembly, but to decompile it, make modifications, then recompile it. In this PDF-eBook, you'll learn about a technique called *Obfuscation*, that can help you avoid this problem. And you'll receive an in depth look at one particular approach to obfuscating your .NET assemblies, **along with a link to a free download of Desaware's open source QND-Obfuscator**.

Regular Expressions with .NET

It might surprise you to know that yet another language is built into Visual Studio – one that can be used in conjunction with VB .NET or any other .NET language. Regular Expressions is an extremely powerful language designed to parse and manipulate blocks of text. Ideal for processing text and validation, understanding Regular Expressions is essential for any .NET developer.

This eBook is intended to be a complete introduction to Regular Expressions that can even be read and understood by programmers who have never heard of them. It is also intended to help experienced Regular Expression programmers come up to speed quickly on the .NET implementation of Regular Expressions.

Tracing and Logging with .NET

The .NET framework includes a powerful tracing and diagnostic system, one that goes far beyond the simple Debug.WriteLine most developers are accustomed to. In this eBook will introduce you to the .NET diagnostic system, teach you tracing and logging design patterns, and demonstrate advanced techniques such as tracing directly into a database or directing trace data directly to an outgoing Email message.

Exploring .NET

Each eBook in this series contains a selection of Dan Appleman's technical articles on .NET. Covering virtually every subject in .NET, and written in a variety of styles, you'll find them entertaining as well ad educational. Refer to the web site for a complete list of articles.

Introduction to NT/2000 Security Programming with Visual Basic 6

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

Also available (print books)

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

This book includes advanced techniques, in-depth explanations of how and why COM/ActiveX components work the way they do, and how to take full advantage of the capabilities of VB6's component and control development features.

Dan Appleman's Visual Basic Programmer's Guide to the Win32 API

This best-selling reference is what the Windows software development kit would look like if it were written for Visual Basic programmers. With over 80 sample programs, it is your best reference to the core Win32 API. Refer to: www.desaware.com for a complete edition history before you upgrade.

Dan Appleman's Win32 API Puzzle Book & Tutorial for VB Programmers

In this book, Appleman teaches you the skills you'll need to use the 7500+ API functions not covered in his famous *Visual Basic Programmer's Guide to the Win32 API*. You'll learn how to interpret the Microsoft documentation and create declarations for even the most difficult functions.