CMP
United Business Media

Graphics & Interfaces

New Column!
Visual C++.NET Expert

JUN
2003

windows developer

APPLICATION DEVELOPMENT FROM WINDOWS TO WEB

NETWORK

SVG—Graphics in XML

Multi-Layered Apps for WinNT/2000

Temporary Object Bug

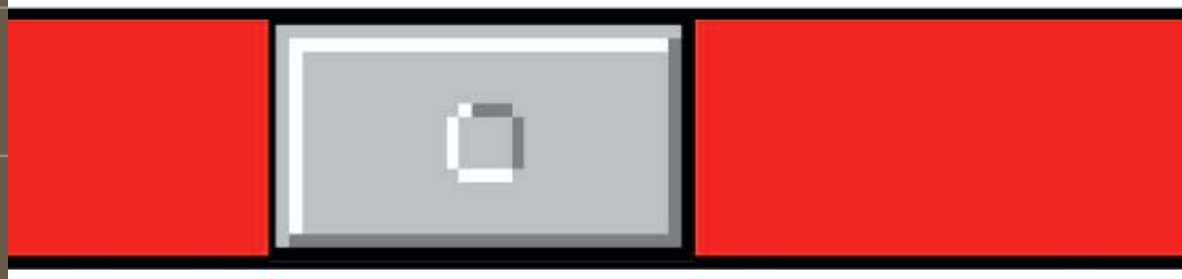
A Better Macro Wrapper for VC++

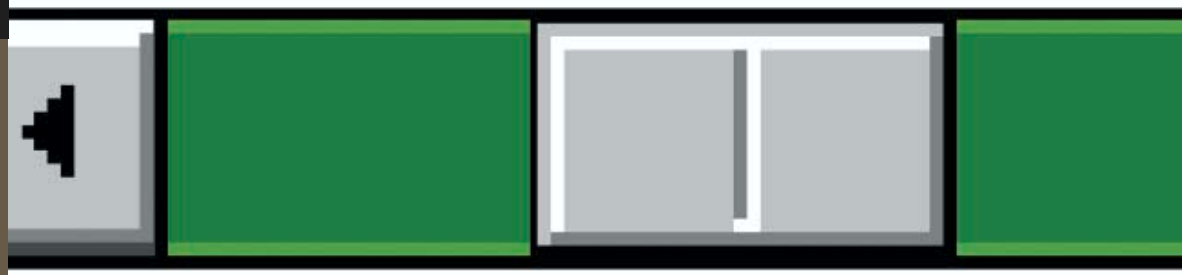
Locking Window Updates

Smarter Merges in ASP.NET

App Domains and Threads in VC++.NET

Build a Custom Scrollbar Control in MFC

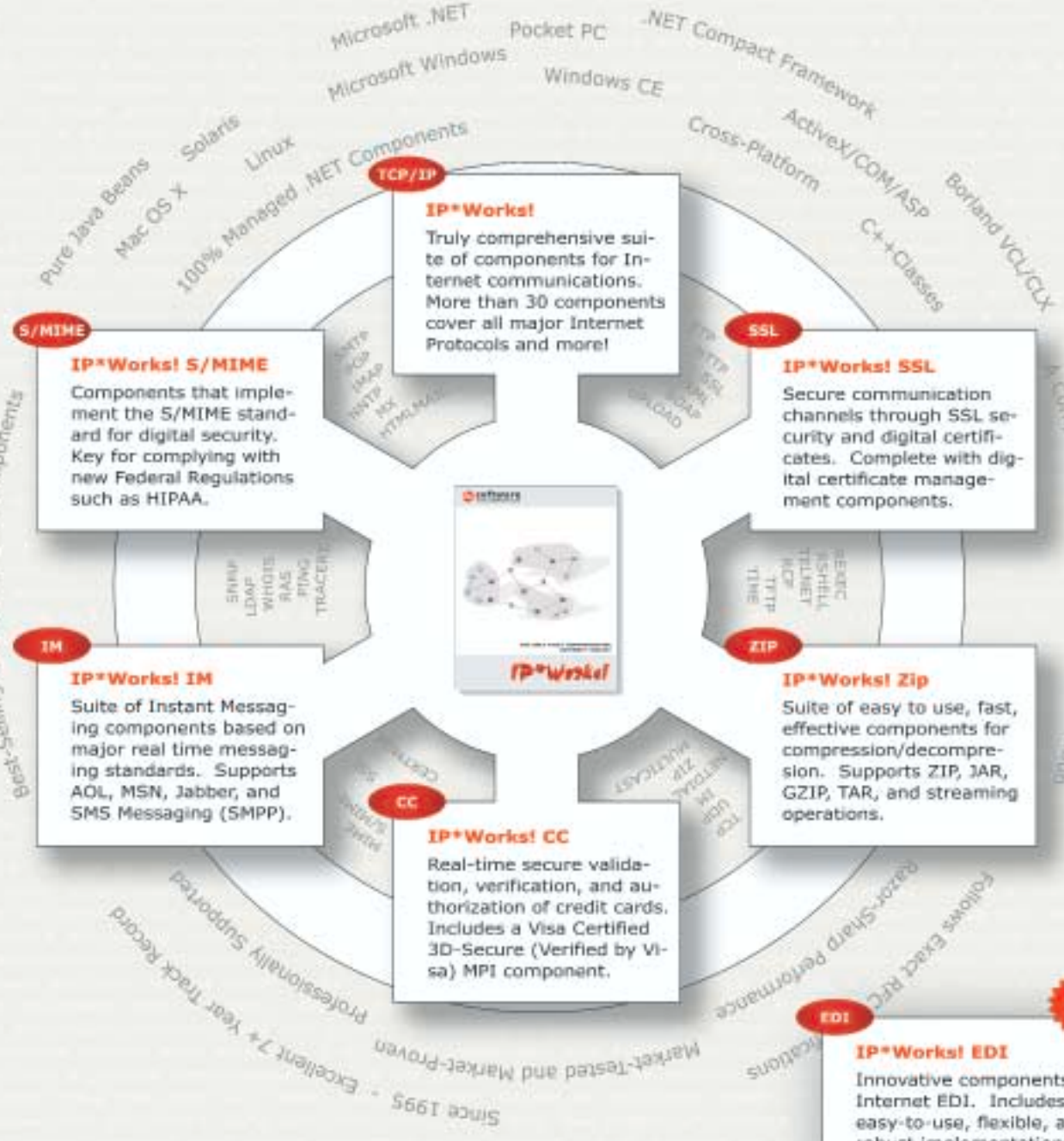




Volume 2 / No. 6

www.windevnet.com

from email to secure credit card transactions, only one component suite does it all!



software

Does your Team do more than just track bugs?

Download Your Free Trial at www.alexcorp.com

Alexsys Team does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

Track all your project tasks ...

- Defects
- New Product Features
- Help Desk Cases
- Action Items
- ISO 9000 Issues and more…

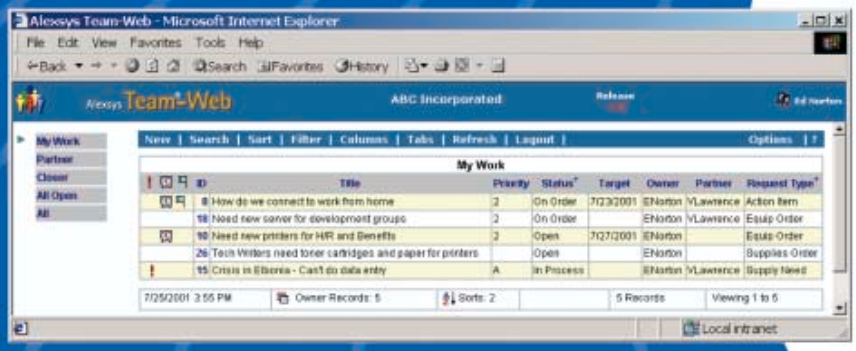
New Version Available Team v2.3

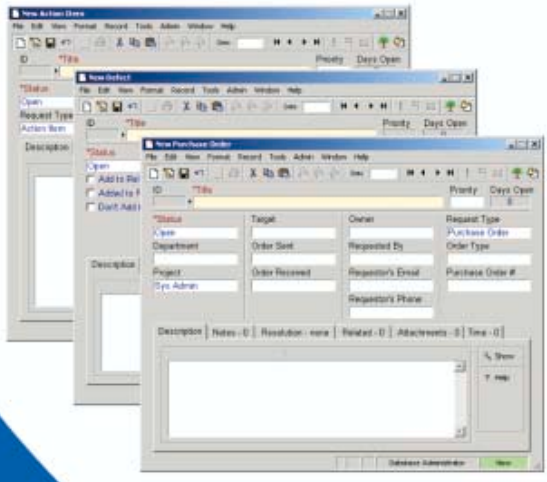
... in one database so you can work together ...

- Keep your project on-track with up-to-date status
- Allow team members to see their assignments and stay on top of the issues
- Find the information you want when you want it
- Run effective meetings using online agendas
- Balance project and team member workloads
- Keep a complete history of your work and more …

... to get projects done.

Alexsys TEAM 2





Team 2 Features:

- Affordable and scalable
- Multiple work request forms to make data entry a breeze
- Automatic Escalations to stay-on-top of critical tasks
- Knowledge base for you and your customers to find answers fast
- Time recording to manage budgets and improve productivity
- Web Access to work where and when you want
- Secure web forms for customers to get help 24/7

ZDNet Editors' Pick

Download a free, no obligation trial version at www.alexcorp.com.
Need more help, give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team-SQL works with Microsoft SQL and Oracle Servers.
All versions of Team 2 work on Windows 95/98/Me and Windows NT/2000/XP, Netscape and Microsoft internet browsers.

windows::developer
APPLICATION DEVELOPMENT FROM WINDOWS TO WEB NETWORK

PUBLISHER Kerry Gates
EDITOR IN CHIEF John Dorsey

EDITORIAL

MANAGING EDITOR **Amy Stephens**

CONTRIBUTING EDITORS **Jeff Claar, Dino Esposito, George Frazier, Richard Grimes, Petter Hesselberg, Paula Tomlinson, Victor R. Volkman**

EDITORIAL ADVISORY BOARD **Mark Baker, Jeff Claar, Dino Esposito, George Frazier, Richard Grimes, Petter Hesselberg, Mark Nelson, Mark Russinovich, Paula Tomlinson, Victor R. Volkman**

ASSOCIATE EDITOR **Della Song**

ART DIRECTOR **Beatriz Américo**

WEBMASTER **Joe Lucca**

SEND READER MAIL TO: **wdletter@cmp.com**

SUBSCRIPTION INQUIRIES: **wdnetwork@halldata.com**

ADVERTISING AND MARKETING

DIRECTOR OF SALES **David Timmons**

REGIONAL MANAGER, EAST
Jon Hampson 603-924-8500 *jhampson@cmp.com*

REGIONAL MANAGER, CENTRAL/SOUTHEAST
Ed Day 785-838-7547 *eday@cmp.com*

REGIONAL MANAGER, WEST
Michele Hurabiell 415-947-6199 *mhurabiell@cmp.com*

ACCOUNT MANAGER, ALL REGIONS
Julie Thibault 603-924-8400 *jthibault@cmp.com*

PRODUCTION COORDINATOR
Michael Penne *mpenne@cmp.com*

DIRECTOR OF MARKETING **Karen Tom**

CIRCULATION

SENIOR CIRCULATION MANAGER **Cherilyn Olmsted**

ASSISTANT CIRCULATION MANAGER **Gwen Olson**

SUBSCRIPTIONS: Annual renewable print subscriptions to *Windows Developer Network* are $34.99 U.S., $45 Canada and Mexico, $65 elsewhere. Payments must be made in U.S. dollars. Make checks payable to *Windows Developer Network*.

CUSTOMER SERVICE: For subscription orders and questions, contact lawrencecs@cmp.com.

ADVERTISING: For rate cards or other information on placing advertising in *Windows Developer Network*, contact the advertising department at 785-841-1631, or write *Windows Developer Network*, 1601 W. 23rd St., Suite 200, Lawrence, KS 66046-2700 USA.

Windows Developer Network (ISSN 1543-6462) is published monthly by CMP Media LLC, 600 Harrison St., San Francisco, CA 94107 USA, 415-947-6000.

CMP MEDIA LLC

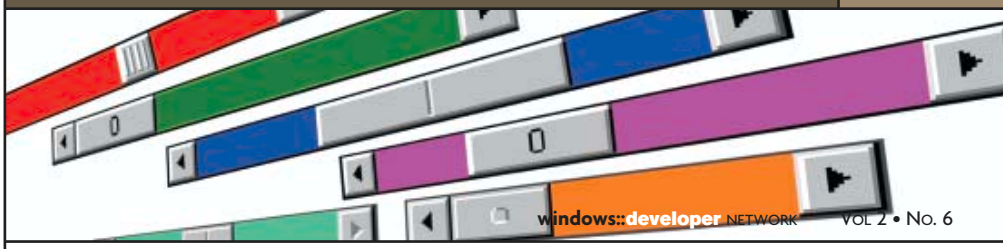
PRESIDENT AND CEO **Gary Marshall**
EXECUTIVE VICE PRESIDENT AND CFO **John Day**
EXECUTIVE VICE PRESIDENT AND COO **Steve Weitzner**
CHIEF INFORMATION OFFICER **Mike Mikos**
EXECUTIVE V.P., CORPORATE SALES AND MARKETING **Jeff Patterson**
PRESIDENT, TECHNOLOGY SOLUTIONS GROUP **Robert Faletra**
PRESIDENT, HEALTHCARE GROUP **Vicki Masseria**
SENIOR V.P., H.R. AND COMMUNICATIONS **Leah Landro**
VICE PRESIDENT AND GENERAL COUNSEL **Sandra Grayson**
GROUP PUBLISHER, ELECTRONICS **Paul Miller**

CMP MEDIA MANAGEMENT

V.P. SOFTWARE DEVELOPMENT MEDIA GROUP **Peter Westerman**

CONTENTS

JUN 2003



FEATURES

COLUMNS

VISIT US ONLINE: www.windevnet.com

DEPARTMENTS

PDF EXTRAS

Download the PDF version of this month's issue to access bonus features. This content includes:

- Listing and figure for "Developing a Custom Scrollbar in MFC"
- Listing for "Rendering Graphics with SVG"
- Listings and figures for "Managing Multi-Layered Apps"

| Download code › **windevnet.com/wdn/code/** |

COMING NEXT MONTH: **WEB SERVICES**

CMP United Business Media

FROM THE EDITOR

MICROSOFT'S DUAL LAUNCH OF Windows Server 2003 and Visual Studio .NET 2003 in April might not have had as big an impact as the launch of, say, Windows 95. Throngs of diehard fans didn't camp out in front of OfficeMax at midnight to buy the latest upgrade. Richard Burte, product manager for the Developer Platform and Evangelism Division at Microsoft, described the new version of VS.NET as a "Fit-and-Finish" release. Nevertheless, the bug fixes and other improvements in both products will be welcomed by developers. C++ programmers in particular have some compelling reasons to upgrade, including the stronger C++ Standards conformance and the Windows Forms designer.

The focus of last years' launch of VS.NET 1.0 was primarily web services and C#, and just days before the launch of VS.NET 2003, the research firm Doculabs (http://www.doculabs.com/) released a Web Services Performance study that showed the .NET 1.1 Framework holding an advantage over three different Java 2EE-based platforms. PR Newswire provided a summary of the study: http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=SVBIZINK3.story&STORY=/www/ story/04-22-2003/0001930511&EDATE=TUE+Apr+22+2003,+08:23+AM

The focus of this year's release was not on web services but rather on improvements in security and reliability. For example, in new VS.NET C++ projects, the /GS flag will be on by default. Also, C++ projects can have a random "canary" inserted on the stack to catch the most-common buffer overruns.

The Visual Studio .NET 2003 release was focused not on web services but rather on security and reliability

But compiler switches and the like are just a part of building more secure apps. Microsoft's Patterns and Practices site (http://msdn.microsoft.com/practices/) provides continually updated recommendations and examples for making your apps as secure and reliable as possible so you can stay ahead of the latest hacks. VS.NET 2003 will also benefit from security checks that Microsoft's development teams have performed on a variety of libraries, including the STL, ATL, MFC, and C Runtime Libraries.

With these improvements to C++ in VS.NET 2003, there will likely be a lot of important topics to cover in the coming months, so we're launching a new column, the "VC++.NET Expert," written by Contributing Editor Richard Grimes. Richard has been covering a variety of .NET topics in our .NET Newsletter; now in the "VC++.NET Expert" column, Richard will give you insights into getting the most out of VC++.NET while avoiding some of the pitfalls of integrating .NET and legacy code. His inaugural column in this month's issue discusses writing an app that has multiple application domains. The "VC++.NET Expert" will appear every other month.

Richard will continue to write the bimonthly .NET Newsletter as well. If you have not signed up for it yet, visit http://www.windevnet.com/newsletters/. You can also subscribe to our other newsletters including Windows Q&A, by Mark Baker, and Dino Esposito's ASP.NET2theMax.

One final note: Regardless of what tools you use for development—VS.NET, C++Builder, Codewarrior, Java, Delphi, and so on—if you've got a helpful coding tip, a simple workaround, or just a caveat to share, send them to our Tech Tips Editor, George Frazier, at wdletter@cmp.com with "Tech Tip" in the subject line. You'll receive $50.00 or more for each tip published.

John Dorsey
Editor in Chief
wdeditor@cmp.com

GAIN THE ADVANTAGE

For over 16 years, SD has delivered reliable, trustworthy, and practical developer training to thousands of developers and managers seeking the best education in C++, Java, XML, Windows, Web Services, modeling, and development process. From the producers of SD, comes a new conference - SD Best Practices.

CONFERENCE TRACKS

- Requirements & Analysis
- Testing and Quality
- Build and Deploy
- Design & Architecture
- Process and Methods
- People, Projects and Teams

YOU WILL GAIN

- Practical skills on how to incorporate best practices, quality design, and proven management techniques into your software applications
- The benefits of modeling, requirements, analysis, and testing
- A deep understanding of Agile, XP and other key methodologies
- Management tools to help you increase team productivity

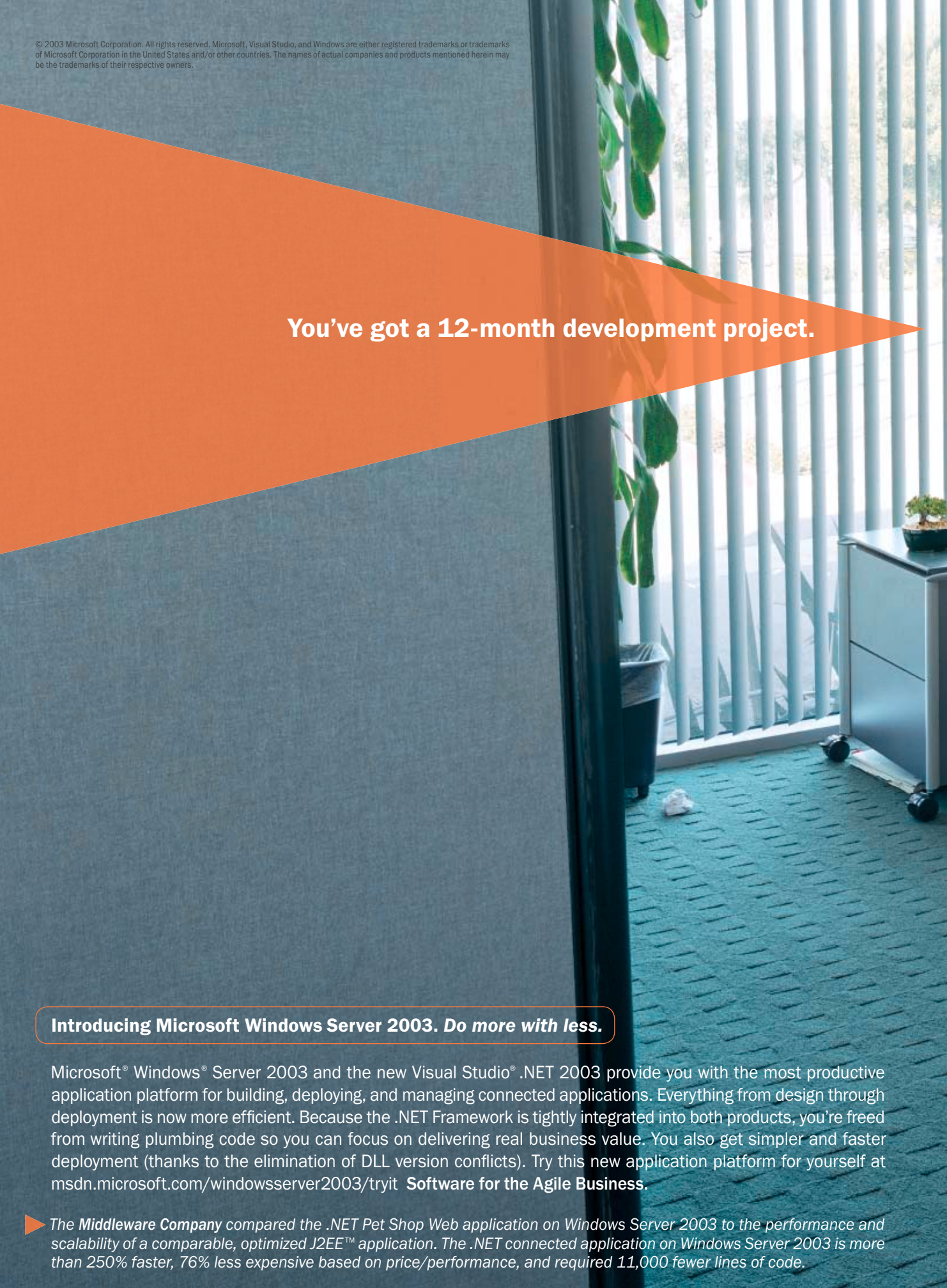
SD 2003 BEST PRACTICES

CONFERENCE & EXPO

A New Conference from the **SD Events Group**

September 15-19, 2003, Boston, MA

Vlisit www.sdexpo.com for details

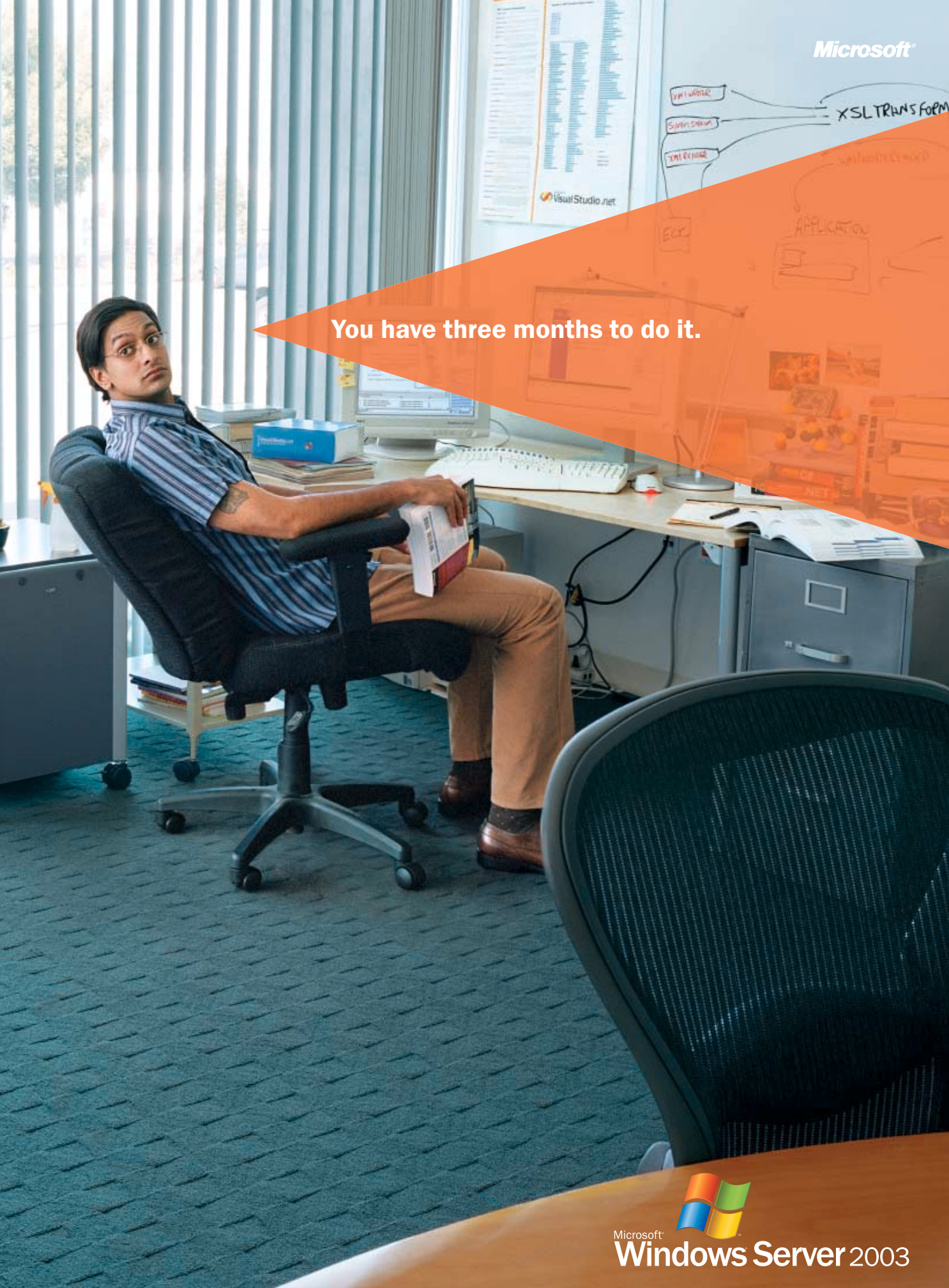


You've got a 12-month development project.

Introducing Microsoft Windows Server 2003. *Do more with less.*

Microsoft® Windows® Server 2003 and the new Visual Studio® .NET 2003 provide you with the most productive application platform for building, deploying, and managing connected applications. Everything from design through deployment is now more efficient. Because the .NET Framework is tightly integrated into both products, you're freed from writing plumbing code so you can focus on delivering real business value. You also get simpler and faster deployment (thanks to the elimination of DLL version conflicts). Try this new application platform for yourself at msdn.microsoft.com/windowsserver2003/tryit **Software for the Agile Business.**

*The **Middleware Company** compared the .NET Pet Shop Web application on Windows Server 2003 to the performance and scalability of a comparable, optimized J2EE™ application. The .NET connected application on Windows Server 2003 is more than 250% faster, 76% less expensive based on price/performance, and required 11,000 fewer lines of code.*


Microsoft
XSLTRANSFORM
VisualStudio.net
You have three months to do it.
Microsoft
Windows Server 2003

DON METZLER

Developing a Custom Windows Scrollbar Control in MFC

Gain complete control over all drawing, mouse, and functional operations

SCROLLBAR CONTROLS ARE USEFUL for many functions. They can be used to select a value, scroll a window, or set a zoom range. They are typically used for scrolling window or drawing areas, as setting position data is typically done with the trackbar (slider) controls. And scrollbars still seem to have their purpose in Windows. However, would they be more useful if they could be customized?

One problem with using a standard scrollbar in Windows is that you have to handle the WM_HSCROLL and/or WM_VSCROLL messages. These messages are handled in the scrollbar's parent window, and then need to handle the scrollbar position and range to be able to have the parent window update the scrollbar position. So you can see that the scrollbar object itself is not really in control of updating its value and position. The trackbar (slider) control in Windows does handle updating its position and then notifying the parent of the position change (also with the WM_HSCROLL/WM_VSCROLL messages). Wouldn't it be nice if the scrollbar behaved more like the trackbar control? While we are talking about limitations with the scrollbars, I have noticed that their range is still limited to integer values. (What year is this?) So if we are going to create a custom control, we should probably extend the range to LONG values.

As a developer, you know that scrollbars do not support owner drawing, such as buttons or listboxes. There is also no provision for custom drawing like a list or tree control. As a matter of fact, the scrollbar controls in Windows are pretty much untouchable as far as customization (other than changing the background color by handling the WM_CTLCOLOR message). So what are the options? You could have some fancy subclassing, but why not just write a custom scrollbar control and have complete control over all drawing, mouse, and functional operations?

When I first thought about this, I, too, put it off due to fear of having to replicate most of the code that already existed for a scrollbar. It seems like a daunting task. I searched the Web for existing code, hoping that someone had already written a custom scrollbar replacement. But even with the help of Google, I was unable to find anything for Win32 or MFC. So I sat down and began charting a course through the unknown waters of writing a custom scrollbar control. I knew I would have to support most, if not all, of the functionality and design of the existing scrollbars—horizontal and vertical styles, scroll arrows, thumb scroller, and so on. But as I started jotting down notes, I realized that there were only a few basic things that needed to be handled: drawing, hit-testing, and mouse movement. That sounds like an over-simplification, but it really is the case. And to have any of this happen, the first requirement was to map out the components (and thus, rectangle areas) of a scrollbar.

Custom Scrollbar Design

Let's determine the requirements for a custom scrollbar control. First, it must look and behave like the scrollbar that users are already familiar with. Second, it should support a range of LONG values. Third, it should support custom drawing. This is a given, as we are creating a control from scratch. If we create the control in a true object-oriented design, the draw methods can each be subclassed/overridden to enable new classes to inherit from the scrollbar and customize drawing. Last but not least, the scrollbar should handle updating itself when its position is changed, and simply notify the parent window when the position does change. So with that, let's put together the design.

Step one—layout the parts of a scrollbar. A scrollbar is a relatively simple control, mostly because it is rectangular. It has many parts, so there is more work in the hit-testing than, for example, a button control, but if you simplify the areas, it becomes manageable. In a horizontal scrollbar (going from left to right in Figure 1), you have the left arrow button for scrolling, the area to the left of the thumb slider, the thumb slider itself, the area to the right of the thumb slider, and finally the right arrow button. Depending on the position of the thumb slider, the page up/down area to the left or right of the thumb may not exist. The thumb slider is the only part of a scrollbar that moves its position, and we can calculate that position based on the scroll range and current position of the thumb.

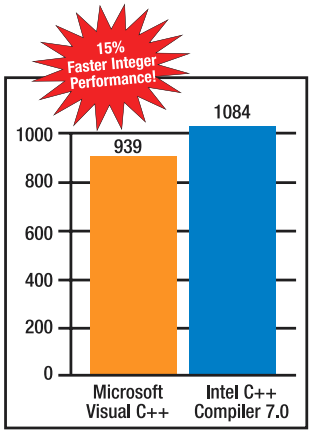
The next step is to enable a response to a mouse-down message and detect which area of the scrollbar was affected. You will see various methods in the code that are named "`IsMouseOver...()`". These helper methods can be used to determine if the mouse is over a specific area in the scrollbar. For instance, `IsMouseOverScrollButton()` can be called with either of the scroll buttons as a parameter to determine if the mouse was clicked (or is over) one of the scroll buttons at either end of the scrollbar.

DON METZLER *is the president and lead software developer for Barefoot Productions Inc. in Louisville, Colorado. He has been developing software since 1980, and has been programming in C/C++/C# Windows application development since 1990. He has assisted more than two-dozen companies in bringing their commercial Windows application software to market. He has also authored two-dozen programs of his own, released as shareware.*

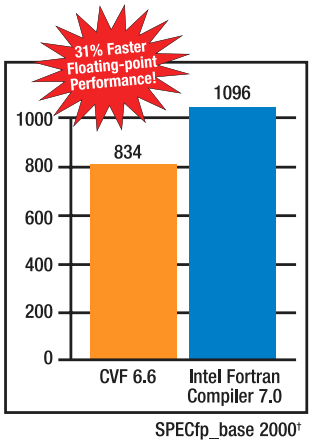
Maximize Your Application Performance

Intel® C++ and Fortran Compilers 7.0 for Windows* and Linux*

Intel's expertise in processors really shows in this latest release of their high-performance C++ and Fortran compilers. Intel compilers provide access to the latest advanced processor technologies. Give your application a performance boost without making source code modifications!



SPECint_base 2000†



SPECfp_base 2000†

† Intel® Pentium 4 Processor, 3.05 GHz, 512 KB L2 Cache, 256MB Memory, Windows XP Professional MP Kernel, Build 2600

Performance

Outstanding performance on Intel architecture including Intel® Pentium® 4, Intel® Xeon™ and Intel Itanium® 2 processors, as well as support for Hyper-Threading Technology.

Compatibility

C++

- Plugs into Microsoft Visual Studio* (6.0 and .NET*)
- Native source and binary compatibility with MS Visual C++*

Fortran

- Strong compatibility with Compaq Visual Fortran*

Support

Purchase includes 1 year of free product upgrades and Intel Premier Support

"The Intel Fortran Compiler 7.0 for Windows is a worthy successor to Compaq Visual Fortran and...remains head and shoulders above its competitors in both features and quality of implementation. This is a fast, efficient, and powerful compiler—a must-have tool for any Twenty-First Century Fortran migration or software development project."

—Dr. Robert R. Trippi
Visiting Scholar/Professor, Computational Finance
University of California, San Diego

YOU SAVE UP TO $163!	Paradise #	Retail	Discount
Intel® C++ Compiler 7.0 for Windows**	I23 0A10	$399.00	$276.99
Intel® Fortran Compiler 7.0 for Windows**	I23 0A11	$499.00	$335.99

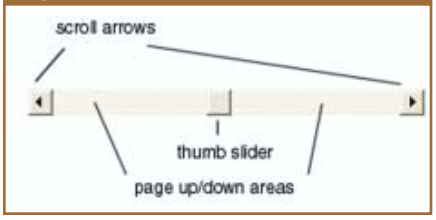
** *Also available for Linux*

FREE 30-day trials available at:
programmersparadise.com/intel/wdj

Programmer's Paradise®
To order or request additional information call:
800-423-9990
Email: intel@programmers.com

03CT06F

Figure 1 Parts of a scrollbar



The code for the custom scrollbar class (CScrollBarEx, for scrollbar extended) is in the ScrollBarEx.h and ScrollBarEx.cpp files. The class is inherited from a standard CWnd class, which means our class will be able to handle all of its own mouse and drawing operations.

Full source code is provided for the custom scrollbar, as well as sample code to show various custom scrollbars created in a dialog, as shown in Figure 2 (available online). Note that although the sample dialog and program only show examples of horizontal scrollbars, the CScrollBarEx class does support vertical scrollbars as well. The source code for this article is available for download online.

Drawing the Scrollbar

Now that the areas have been defined, each part can be drawn. The CScrollBarEx::Draw() method handles the drawing. In turn, this method will call other draw methods to handle each part of the control: the background, scroll areas, arrow buttons, and thumb slider. Note that in the Draw() method, CMemDC is used for the drawing DC. This is an extended CDC object so that drawing can occur offscreen and then be blitted to the screen without flicker. DrawBackground simply fills the entire client rectangle with the background color.

After drawing the background, DrawScrollAreas() is called to handle drawing the area(s) between the arrow buttons and the thumb slider. If the thumb slider is at the minimum or maximum position, there will only be one scroll area to draw. Note that the scroll areas will be drawn, inverting the background pixels, if the mouse is being used to scroll by the page size in the scroll area. If not, the scroll area will simply remain the background area drawing in the DrawBackground() method.

The arrows at either end of the scrollbar can be drawn with the Win32 API call: DrawFrameControl(). This function is useful for a wide variety of drawing controls in Windows. (If you look at the MSDN documentation for DrawFrameControl, you will see just how powerful it is for drawing.) The scrollbar uses it to draw the arrow buttons at the ends of the scrollbar. The DrawButtons() method handles drawing the buttons in our custom class.

To complete the scrollbar drawing, DrawThumbSlider() handles the drawing of the thumb slider. After that, the CMemDC is released, which copies all of the offscreen drawing to the screen in the scrollbar's client rectangle.

The Windows scrollbar also uses a visual indicator to show when a scrollbar has focus. It does not appear often, as scrollbars are typically used as controls on other windows. In this case, the window itself has focus, so the scrollbars will not show their focus state. However, in the case of standalone scrollbar controls, when a scrollbar gets focus, you will see that the thumb slider "flashes" to indicate the control indeed has focus. This is handled in our custom scrollbar control using a timer, which gets set when the WM_SETFOCUS message is received. An internal flag is set, and when the timer is triggered (TIMER_FLASHFOCUS, on 500 ms intervals), the flash flag is toggled and the scrollbar is redrawn. The resulting effect is that the thumb slider is flashed from a background system color of COLOR_3DSHADOW to COLOR_3DFACE.

Mouse Interaction

Now that the parts have been drawn, we need to make them interactive with the mouse. This is not all that difficult, once you know the rectangular areas you have defined. Basically, you will have to track various mouse states:

- When and where the mouse is clicked in the scrollbar (WM_LBUTTONDOWN).
- When the mouse is moving (WM_MOUSEMOVE) and how it affects the scrollbar position and drawing.
- When the mouse button is released (WM_LBUTTONUP), stop all actions on the scrollbar and return to an idle state.

When the control receives a WM_LBUTTONDOWN message, it is a matter of determining the hit-test area beneath the mouse cursor. There are up to five areas that we must check to determine mouse position. They are the same areas that were outlined earlier in the design section: either scroll button, scroll area (to either side of the thumb slider), and the area of the thumb slider. We will store this area in m_ScrollBarClickArea, as the mouse move handling will depend on what area/action is occurring.

When the mouse button is down, you will actively track the mouse position by calling SetCapture(). This will ensure that all WM_MOUSEMOVE messages go to the control, even when the mouse cursor is not over the control. To ensure Windows does not get into an improper state, we need to be sure and call ReleaseCapture() when the WM_LBUTTONUP message is received by the control. So between clicking in the scrollbar and releasing the left mouse button, all the mouse movement will be sent to the control. When the scrollbar thumb is moved, the position is updated accordingly. When the scrollbar position is changed, the WM_SCROLLBARPOSCHANGE custom message is sent to the parent window (WPARAM = the control ID and LPARAM = the scroll position). This message to the parent is simply a notification—the parent does not need to act on it like the WM_HSCROLL/WM_VSCROLL messages.

Clicking on one of the arrow buttons moves the position of the scrollbar by the line size. For scrollbars with large ranges, the scrollbar may not show the position differently, at least visually. (If the range is greater than the number of pixels in the scrollbar, then there are not enough pixels to show each position update.) When an arrow button is pressed, EnableScrollTimer() is called to enable the scrollbar timer. If the mouse is held down on an arrow button, after a short delay, the timer is used to repeatedly scroll the scrollbar until the mouse is released. This mimics the standard scrollbar functionality, which behaves the same way.

Clicking in the scroll area between the thumb slider and arrow button is handled the same way as the scroll arrow buttons, but the scrollbar position is moved by the m_PageSize value rather than the m_LineSize value.

You will also notice that the CScrollBarEx class supports the ::EnableScrollBar() method, which is the same as the CScrollBar::EnableScrollBar() method. This allows either, both, or neither of the scrollbar arrow buttons to become disabled. It does not disable the thumb slider or scrollbar, but only affects the buttons at the ends of the scrollbar.

Keyboard Control

In order for the custom scrollbar to handle keyboard control, it must handle the WM_KEYDOWN message in the OnKeyDown() method. The arrow keys are supported, as are the PageUp/PageDown and Home/End keys. This will work if the scrollbar has the current focus and will process the key presses. Or will it? If the scrollbar control is placed in a dialog, we must also remember to handle the WM_GETDLGCODE message and return a value of DLCG_WANTARROWS from the OnGetDlgCode() method. This will allow the arrow keys to be passed to the scrollbar control rather than being handled by the dialog as navigation keys.

Proportional Thumb Slider

The concept of a proportional thumb slider is not difficult. With some basic math, using a proportional thumb slider is just as easy as using a fixed-size thumb slider. If you look at the CScrollBarEx::GetThumbSliderRect() method, this is used to calculate the bounding

Programmer's Paradise®

Your best source for software development tools!

GUARANTEED BEST PRICES*

Should you see one of these products listed at a lower price in another ad in this magazine, CALL US! **We'll beat the price**, and still offer our same quality service and support!

*Terms of the offer:

- Offer good through June 30, 2003
- Applicable to pricing on current versions of software listed
- Limit: lowest price only
- Offer does not apply towards obvious errors in competitors' ads
- Subject to same terms and conditions

Prices subject to change. Not responsible for typographical errors.

xmlspy® 5

by Altova

xmlspy® 5 is the industry standard XML Development Environment for designing, editing and debugging enterprise-class applications involving XML, XML Schema, XSL/ XSLT, SOAP, WSDL and Web Services technologies. It is the ultimate productivity enhancer for J2EE, .NET and database developers.

Enterprise Edition	Professional Edition
Paradise #	Paradise #
IOD 0166	IOD 0160
$965.99	$389.99

www.programmersparadise.com/altova

Programmer's Paradise #1 Best-Selling Help Authoring Tool for 7 Years Running!

RoboHelp Office

The Industry Standard in Help Authoring

Create professional Help systems for Windows and Web-based applications, including .NET.

- Create all popular Help formats
- Create standard and advanced Help-specific features
- Work in WYSIWYG or true code
- Easily create context-sensitive Help
- Generate printed documentation
- Winner of 55 industry awards

Working in .NET? Effortlessly create your Help system with RoboHelp Office Pro for .NET!

Paradise # E75 02AV

$948.99

www.programmersparadise.com/ehelp

LEADTOOLS Document Imaging

by LEAD Technologies

Document imaging including annotations, specialized bitonal (b/w) image display and processing like scale-to-gray and finer black, performance and memory optimizations for bitonal images, image clean-up like hole-punch, line and staple removal, high-speed scanning.

Paradise # L05 048V

$1,639.99

www.programmersparadise.com/lead

Intel® C++ and Fortran Compilers

by Intel

Increase the Performance of Your Application with Intel's High-Performance Compilers

Version 7.0

Intel's expertise in processors shows in this latest release of its flagship compiler product-line. Version 7 of its C++ and Fortran Compilers for Windows and Linux. Intel Compilers deliver outstanding performance on Pentium® 4 and Intel® Xeon™ processors, and the 64-bit Itanium and Itanium 2 processors and take advantage of Multi-processor systems and Hyper-Threading technology.

Intel C++ for Windows Paradise # I23 0A10

$305.99

www.programmersparadise.com/intel

NEW! Interface Support!

c-tree Plus®

by FairCom

With unparalleled sophistication, c-tree Plus gives developers absolute control over their file management needs. c-tree Plus has established itself as the premier tool for commercial database development. Use low-level routines, high-level ISAM routines, or ODBC/Crystal Report layers for high-speed random or sequential data access. c-tree Plus has been ported to over 100 environments and includes a robust database Server SDK. Develop single-user and multi-user non-server applications or client-side applications for FairCom's robust database server—the c-tree Server. Windows to Mac to Unix all in one package.

Paradise # F01 0131

$850.99

www.programmersparadise.com/faircom

TX Text Control ActiveX 10.0

by The Imaging Source

Add RTF, DOC, HTML, CSS and PDF Support to Your Application

NEW .NET VERSION!

TX Text Control is royalty-free, robust and powerful word processing software in reusable component form. The new Enterprise XML version features a rich set of properties for the manipulation of XML and CSS. Developers can now offer end-users the ability to separate their textual content from their formatting rules.

Download a demo today.

DEMO AVAILABLE

programmersparadise.com

Enterprise Edition	Professional Edition
Paradise #	Paradise #
T79 0214	T79 0215
$1,495.99	$729.99

www.programmersparadise.com/theimagingsource

TeeChart for .NET Control

by Steema Software

100% managed C# Net charting component with optional full C# source code available. For desktop, server based and ASP.Net applications. Extensive demos for Visual Studio .NET including WinForms and ASP.Net WebForms. Database enabled. Multiple export formats. Multiple axes. Multiple Chart types in 2D, 3D and OpenGL. Custom Canvas. Grid, Toolbar and many specialist financial functions including MACD, ADX and Stochastic.

Paradise # SOT 0130

$474.99

www.programmersparadise.com/steema

PR-Tracker™ v5.1

by Softwise Company

Affordable scalable enterprise level bug tracking system featuring classification, assignment, sorting, searching, reporting, access control, user permissions, attachments and email notification. Integrates with PR-Tracker Web Client (included) and ProblemReport.ini (included for your betatest or customer support interface). Supports Access and SQL Server.

Download Today!

DOWNLOAD NOW!

programmersparadise.com

Paradise # S3R 0147

$117.99

www.programmersparadise.com/softwise

Paradise Picks

DevTrack 5.1

by TechExcel

The only defect tracker with a built-in web portal for your Beta customers. DevTrack 5.1 comprehensively manages and automates development management, defect tracking, and test plan management processes within one seamlessly integrated application. It provides powerful workflow and process automation features, seamless source code control integration with Microsoft Visual SourceSafe (VSS) and Perforce, robust searching and reporting, and comprehensive point-and-click customization.

programmersparadise.com/techexcel

Paradise # T34 0194

$482.99

Sun™ ONE Studio 7 Enterprise Edition Solaris

by Sun Microsystems

Sun™ ONE Studio 7, Enterprise Edition for Solaris is a productive environment for developing reliable, scalable, high-performance applications in the C, C++, Fortran, and Java languages for the Solaris Operating Environment. Sun ONE Studio 7, Enterprise Edition for Solaris software is a bundle that includes the Sun ONE Studio 7, Compiler Collection and the Sun ONE Studio 4, Enterprise Edition for Java products.

www.programmersparadise.com/sunone

Paradise # S69 0U58

$2,890.99

800-445-7899 • programmersparadise.com

rectangle of the thumb slider. This calls `GetThumbSliderWidth()`, which computes the current thumb slider width, depending on the value of the `m_ThumbSliderSize` variable. This could be one of the following states: THUMBSLIDERSIZE_DEFAULT (the standard size of a thumb slider, using GetSystemMetrics(SM_CXHTHUMB)), THUMBSLIDERSIZE_FIXEDPIXELS (a fixed thumb slider size), or THUMBSLIDERSIZE_RELATIVEPCT (which is the proportional thumb slider).

Customization Methods

Various functions have been added to show how you may customize the scrollbar code. The following methods in the `CScrollBarEx` code enable some customizations:

```
SetThumbSliderSize(EThumbSliderSize
    inThumbSliderSize,
    short inThumbSliderSizeValue)
// For inThumbSliderSize:
// THUMBSLIDERSIZE_DEFAULT,
//    inThumbSliderSizeValue = IGNORED
// THUMBSLIDERSIZE_FIXEDPIXELS,
//    inThumbSliderSizeValue = pixels
// THUMBSLIDERSIZE_RELATIVEPCT,
//    inThumbSliderSizeValue = percent
//    of total scrollbar pixel size

SetThumbStyle(EScrollThumbStyle
    inThumbStyle) : Sets the thumb style

SetBackgroundColor(COLORREF
    inBackgroundColor): Sets the
    background color of the scrollbar.

SetHotTrackThumb(BOOL bHotTrackThumb) :
    Sets the hot-tracking ability
    for the thumb slider.
```

Future Improvements

Now that I have presented the basics for a custom scrollbar control, here are some thoughts for improvement. First, it could be enhanced to use the XP Theme Manager to make the control look like the standard Windows XP scrollbars.

You could add additional buttons to the end(s) of the scrollbar (next to the arrows), similar to the scrollbars used in the MS Office applications. Maybe you want to change the background from a filled color to an image or pattern. You could subclass a window's existing scrollbar(s) with a `CScrollBarEx` instance to make them customized. And now that you have control over the entire scrollbar, you may want to customize the cursors to something other than the standard arrow. Finally, porting this code to C# and using GDI+ should not be too tough a task, and may provide good learning experience for those wanting to get into .NET coding.

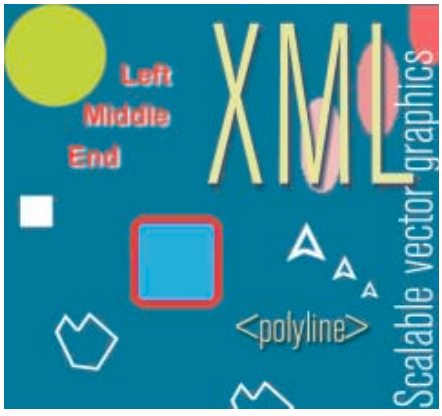
The possibilities for customization are endless. However, I hope that what I have presented here has taught something about writing a custom control and can perhaps get the sparks of innovation flying for some cool customizations. Please send me an e-mail—I'd like to hear about your ideas. **w::d**

| Download code › **windevnet.com/wdn/code/** |

Table of Contents

Rendering Graphics with SVG

The Scalable Vector Graphics spec lets you define graphics and animation in XML



THERE IS NO DOUBT that the popularity of XML is continuing to grow. Through the availability of namespaces, XML can be used to represent a number of defined languages, most of them relating back to XML, such as XSLT, XSL-FO, and XSD. Additionally, XSLT has added a lot of value to XML by making it possible to generate documents on the fly, so it's not surprising to find that W3C has created another XML-based language, SVG, which can be used to present graphical output. SVG stands for "Scalable Vector Graphics," which is somewhat of a tautology, in that scalability has always been one of the advantages of vector graphics.

SVG is a relatively new technology, and no browser currently supports it natively. Instead, a plug-in is required, the most popular being Adobe's; see http://www.adobe.com/svg/viewer/install/main.html. Once you have installed the viewer and have downloaded the code for this article, open SVGDemo.htm in your browser and you'll see something like Figure 1. This page shows off the main SVG elements, which I'll describe, but first I want to note some basic points about the syntax. First of all, it is quite obviously valid XML. More importantly, being an XML syntax, it is case sensitive, so we will discuss case matters with regard to all the attributes and node names. The full SVG code for this example is in Listing 1 (available online).

Defining Shapes, Text, and Animation

The first few lines simply define the location of the DTD for SVG, and the size of the rendering space. The next element is a node called <defs>. The elements found within this one are definitions; they are not rendered, but are reused later.

After this, we have our first shape, a circle. The properties of the circle are provided using attributes of the node, and the minimal set is used here; the center as two points, a radius, and a fill color. The same is true of the next node, which is a rectangle and is defined in terms of top-left points, width, and height. The second <rect> node introduces `rx` and `ry`, which allow a defined degree of rounding to the rectangle. Additionally, it introduces the stroke and stroke-width attributes. These can be applied to all geometric shapes and define the color and width of the pen that draws the outline. The final shape is an <ellipse>, which also introduces the "fill-opacity" attribute. Both the fill and the stroke have an optional opacity attribute, which can be set to anything from 0 (invisible) to 1 (fully drawn). Any value in between defines the degree to which the item is drawn, and the degree to which the underlying items show through.

As well as these basic shapes, we are able to define our own by providing a series of points. The first element to do this is <polyline>, which simply accepts a group of points and draws lines between them. Both this element and <polygon> have a "fill" attribute, which specifies the color with which to fill the shape. The difference between the two is that <polygon> draws a closing line from the last point to the first. In the example, both elements have the same points to illustrate this difference.

The final example here is a path, which is a highly flexible way of defining a shape. The syntax takes a series of points, as well as letters to define the nature of each point. They always start with an M, which stands for "move," and then a combination of the letter codes shown in Table 1. Additionally, all of these options can be used with a lowercase letter, the difference being that uppercase provides absolute coordinates, and lowercase provides relative coordinates.

The next examples show how to use the nodes in the <def> tag. Items defined there must have an "id" attribute, and can then be used with the following syntax:

```
<use xlink:href="#id"/>
```

Any parameters can be modified for this instance of the object by specifying them. In this manner, the first two nodes draw two instances of our rectangle, but with different colors and in different positions.

As well as defining shapes by drawing lines, any number of elements can be grouped into a single entity with the <g> node. Such a group is shown in the <def> node and is named "lollipop."

SVG also provides facilities for the display of text, as is shown in the next examples. The only one that needs much explanation is the last, which draws a line from a <def>, then uses the same path in a <textpath> node nested within a <text> node. This has the effect of causing the text to be drawn so that it follows the path. Any text that does not fit on the path is not displayed.

As well as offering these methods to display static items, SVG offers a number of ways to

CHRISTIAN GRAUS *has been programming since 1985. He is especially passionate about the C++ Standard Library, but is also interested in other languages, such as C#. He can be contacted at christian.graus@dytech.com.au.*

provide animation. The simplest is the <animate> tag, which simply changes an attribute of an item between values indicated, within a specified timeframe. It can contain an xlink:href to the item to animate, or be contained within the node in question. The other possible properties are shown in Table 2.

As well as this general animation element, three specific types of animation elements exist. The first is <animateMotion>, which can take `from` and `to` values, or a complex path, as defined earlier. The values listed for specifying durations and repeat counts apply to all animation elements.

Second, `animateColor` allows the color of an element to be changed over time. It uses `attributeName` to specify which color specifier to change, and in that sense seems very similar to just using <animate>; however, <animate> is unable to animate colors. The final animation type is probably the most powerful of all. A node can have a transform attribute, which defines rotation, scaling, and translation (movement in the `x` and `y` planes). While it may seem superfluous to do the following:

```
<rect x="10" y="10"
    width="10" height="10"
    transform="translate(10, 10) scale(4.5)"/>
```

on the basis that the calculated values could just as easily be put into the node in the first place, the power lies in being able to make these transformations on the fly. The <animateTransform> element looks very similar to <animate>, the difference being that we are not changing an entire attribute, as the transform attribute has up to three properties in its value. A sample element may look like this:

```
<animateTransform
    attributeType="XML"
    attributeName="transform"
    type="scale"
    from="1" to="4"
    begin="0s" dur="4s"
    fill="freeze"/>
```

You can use these examples to experiment with the sample SVG in order to get a feel for what can be achieved. However, it is also appropriate that I provide a more complex example. The most obvious example would be one that uses XSL to transform sample XML that conforms to a given schema into some sort of chart, but I felt that this was both obvious and likely to be an easy example to find online or in print. Instead, I have elected to try and see how far SVG can be taken by using it to write an implementation of Atari's classic "Asteroids" game. In order to do this, I have taken advantage of the fact that the SVG engine will continue to render the page according to the XML in it, and this means that we can use the JavaScript DOM to modify that XML in order to achieve animation and other interactive effects. I will not focus much on the JavaScript side of the game, beyond the areas where it directly interacts with the SVG.

SVG and CSS

Listing 2 (available online) shows the SVG portion of the game. The full file contains the script within a CDATA tag. The first thing you may notice is line 3, which points to a CSS file. If you're not familiar with cascading style sheets, they basically provide a way to set attributes of a node in a central location, so that in an HTML document you can specify a particular look by using the syntax "class=classname," rather than specifying a number of attributes every time. This is useful in SVG because it allows us to, for example, have one location in which we can change the way

Figure 1 Interface of SVG Demo

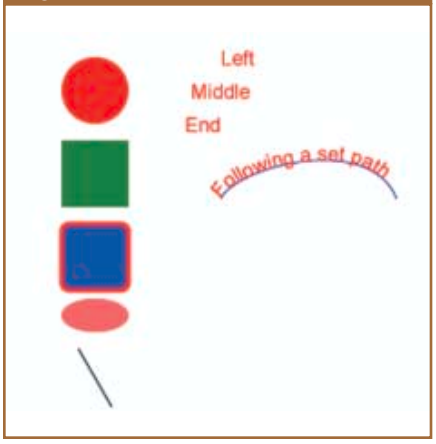


Table 1
Letter codes to define drawing action

Letter	Action	Description
L	Line	Takes point to move to.
H	Horizontal Line	Takes x-coordinate to draw to.
V	Vertical Line	Takes y-coordinate to draw to.
C	Curve	Takes two points, x1y1 and x2y2, and draws a cubic Bèzier curve from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve.
S	Smooth Curve	Takes a point, and extrapolates the other point needed from the previous operation.
Z	Close Path	Always the last parameter, with no params, closes the path by drawing a line to the start point.

all asteroids look. Throughout the game, you'll note that things such as stroke, fill (as a color, as opposed to fill as the last action of an animation), opacity, and so on are never set directly, and this means that if we wanted a more modern look to the game, we could simply set fill colors inside the CSS file instead of having to go right through the document to do it. Here is an example from the CSS file:

```
polygon.asteroid
{
        stroke:rgb(255,255,255);
        stroke-width:1;
        fill:none;
}
```

In contrast to previous examples, the style sheet uses RGB tuples to specify color and the syntax demands that we provide the object type, then the name of the class. This allows us to specify an "asteroid" class separately for different shapes, and it also means we cannot use the aforementioned class within a circle, for example.

Line 4 contains our root SVG element where we also specify an `onload()` handler, which is in line 939. Line 944 contains an item of particular interest:

```
svgDocument =
    event.getCurrentNode().getOwnerDocument();
```

Here we grab the root element of the SVG Document, which provides us with a number of additional methods to the usual DOM calls for manipulating XML.

The first item in the <def> node is a group, used to draw the explosion when the ship hits an asteroid. It has the property "display='none'". There are two ways to control visibility of an SVG element, the first is the display attribute, which can have a number of properties; however, the two most commonly used are "none" and "inline." Display is part of CSS2, and the CSS2 documentation online lists the other 16 possible values, but they are not commonly used in SVG. The other way to control visibility is with the "visible" attribute, which has the following possible values:

- Visible—the current element is visible.
- Hidden or Collapse—the current element is invisible.
- Inherit—the current element will inherit its parent's visibility.

The difference between using these elements is that setting an element's display property always affects its children; children can, however, modify their own visibility, and display='none' removes items from the rendering tree, whereas items are processed regardless of their visibility. Display is therefore the more efficient way to hide SVG items.

Starting on line 13 is an `animateMotion` tag that looks like this:

```
<animateMotion id="Motion1"
    from="0, 0" to="10, -10"
```

```
    begin="undefined"
    onend="NextShip();"
    dur="1s" fill="freeze"/>
```

The two items of special interest here are the `begin` tag and the `onend` tag. The first example only sets `begin` tags, which specify time intervals, but a `begin` tag can specify any combination of time and other SVG events, commonly the beginning of other animations. In this case, however, we have specified the beginning as "undefined", which means this animation will never occur unless we specifically set it into motion ourselves. This occurs in line 705, with the line

```
SvgDocument.getElementById("Motion1").beginElement();
```

The `getElementbyId` method of our document object (which we set in our load handler) gives us an easy, elegant way to reference objects without having to traverse the document ourselves. The `onend` attribute specifies an action to take when the animation ends; in this case, it's a JavaScript method that, among other things, has a similar line that calls `endElement();`. In addition, these two methods set the display attribute with code like this:

```
SvgDocument
    .getElementById("Ship")
    .setAttribute("display", "none");
svgDocument
    .getElementById("Explosion")
    .setAttribute("display", "inherit");
```

In this case, we are hiding the ship and showing the explosion. The reverse occurs in `NextShip();`.

Lines 96–100 specify some empty groups; these are filled by our code and the groups act as a reference point to make it easy to find nodes we have dynamically created. Nodes are created with code like this:

```
var newnode = svgDocument
    .getElementById("BigAsteroid")
    .cloneNode(false);
Asteroids[i].Node = newnode;
Newnode
    .setAttribute ('transform',
     'translate('+Asteroids[i].X+'
    . '+Asteroids[i].Y+')
     rotate('+Asteroids[i].RotAngle+')');
var contents = svgDocument
    .getElementById ('Asteroids');
newnode = contents
    .appendChild (newnode);
```

In this case, we find the node we want, clone it, store it into a JavaScript pseudoclass, set its position, and append the node we've created into the appropriate group. Without the `appendChild` call, our node would not be part of the document. We would remove this node like this:

```
Asteroids[i].Node.parentNode.removeChild(Asteroids[i].Node)
```

The `parentNode` property gives us the proper context to remove the desired child node.

Starting on line 426, the `VectorShot` method tests to see if a particular shot on the screen is within the bounds of a particular asteroid. We first do rect intersection tests on all collision detections in order to perform these expensive operations, if necessary. The point of interest here is that our asteroid node contains an array of points, but they

are the 0,0 centered points that the asteroid started with; all our objects are moved by setting the transform attribute. In order to get a set of points that represents the current position, we need to access this object. But parsing this string to determine the translate coordinates is a little messy, and once we get the rotation, figuring out where that puts the points would be even worse. Luckily, SVG handles all the dirty work for us. Here is a shorter, contrived example to illustrate the methods we use.

```
var ctmAst =
    svgAst.getCTM();
var point =
    svgDocument.documentElement
    .createSVGPoint();
point.x = 10;
point.y = 10;
point = point
    .matrixTransform(ctmAst);
```

The first thing we do is to call the `getCTM` method of an object, which stands for "Current Transformation Matrix." This object represents a six-element matrix, the six elements being named "a" though "f." We then create an `SVGPoint`, which has `x` and `y` properties and a method called `matrixTransform`, which applies a transformation matrix to a point and returns a point with the values after the transformation. By applying this method to all the points in an object, we are able to establish their current location. The `SVGMatrix` also has a number of methods to alter its values as follows:

```
SVGMatrix flipX()
SVGMatrix flipY()
SVGMatrix inverse()
SVGMatrix multiply(SVGMatrix)
SVGMatrix rotate(float angle)
SVGMatrix rotateFromVector(
    float x, float y)
SVGMatrix scale(float factor)
SVGMatrix scaleNonUniform(
    float factorX, float factory)
SVGMatrix skewX(float angle)
SVGMatrix skewY(float angle)
SVGMatrix translate(float x, float y)
```

Additionally, there are `get` and `set` methods for the a–f properties.

Image Processing

As well as all of this vector-based functionality, SVG offers methods to perform image processing. Naturally, in order to process an image, the SVG engine must first render it to a raster image, as image processing is not a vector-based operation, and it is therefore also a costly operation to perform. It would not be wise to use these methods on an interactive page, especially on the changes at the rate a game does. The methods available include `feImage`, `feComponentTransfer`, `feComposite`, `feBlend`, `feFlood`, `feTile`, `feMorphology`, `feConvolveMatrix`, `feDisplacementMap`, and `feTurbulence`. SVG also offers the ability to create elements filled with patterns or gradients. Although the most flexible of these methods is certainly `feConvolveMatrix`, sadly the Adobe plug-in does not support it. I believe this is probably because the specification is ambiguous. A convolution matrix specifies weights

Figure 2
Interface of SVGeroids Demo

Table 2 Animation properties

Name	Description
attributeName	The attribute to animate.
attributeType	Must be XML, or text/CSS (for animating style sheet elements).
begin	When to start animation. Can be a time or reference to another animation event.
dur	How long the animation show lasts.
end	When the animation should end.
repeatCount	Number of times to animate.
repeateDur	Amount of time to repeat for—default is "infinite."
additive	Sum or replace (default) defines how values from different animations on one element interact.
from	Value to start from.
to	Value to move to.
by	Step value.
values	Semicolon-separated list of values to animate between fill freeze to freeze at end, remove to go back to initial state.
onend	Action to take when animation ends.

to give to surrounding pixels in order to calculate the value of a central pixel. Therefore, a 3×3 matrix that looks like this:

```
0 0 0
0 1 0
0 0 0
```

is an "identity" matrix—it produces an exact copy of the original. The SVG specification specifically allows matrices with even numbers of rows and/or columns, which leaves it unclear which element of the matrix represents the pixel being processed. I suspect that Adobe has not implemented this rather useful feature on the basis that it would need to come up with a sensible rule for this eventuality, and in the absence of such a rule, Adobe has opted out. Space does not permit me to discuss these things further. I hope that by providing element names here, and references at the end, I will inspire you to look into these things yourself.

This has been somewhat of a whirlwind tour of SVG, an exciting new way to render interactive graphics for the Web. At this stage, information about SVG is sparse compared to technologies such as Flash, but some of the web sites already online are easily as rich as those created using proprietary formats. It is certainly useful for anyone who is developing a browser-based presentation layer.

References

Scalable Vector Graphics (SVG) 1.0 Specification, http://www.w3.org/TR/SVG/index.html#minitoc.

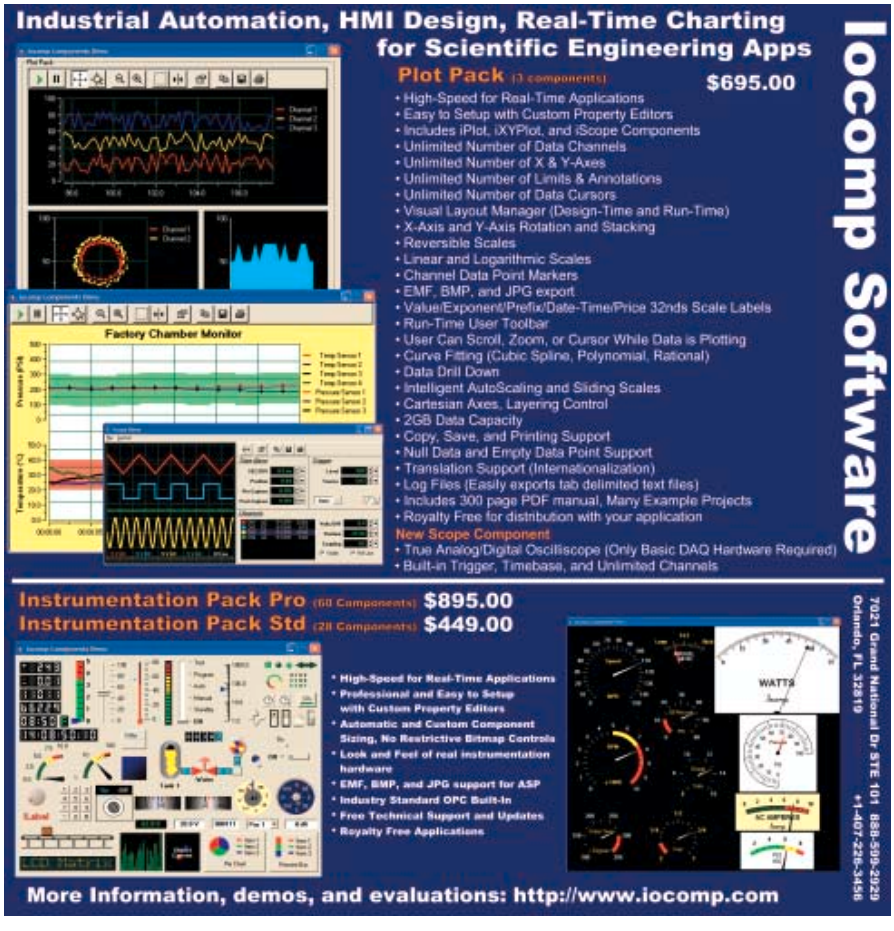
Kevin Lindsey Software Development, http://www.kevlindev.com/.

SVG Essentials, J. David Eisenbery; O'Reilly & Associates, 2002, ISBN 0-596-00223-8.

SVG Viewer plug-in, http://www.adobe.com/svg/viewer/install/main.html.

SVG Blog, http://www.svg.org/. w::d

| Download code › **windevnet.com/wdn/code/** |



Managing Multi-Layered Apps

Tips on developing a shared Windows NT/2000 service

IN A TYPICAL MEDICAL or semiconductor industry scenario, a sophisticated instrument application runs on a dedicated Windows NT/2000 workgroup or domain and is utilized by multiple users. In the most common setup I've seen, the application runs under a single dedicated Windows NT/2000 user account and various users share a single Windows user account. However, this arrangement doesn't protect one user's data from another user's data; network access is also restricted, since different users need to access network resources using their own credentials. It wouldn't be practical for each user to map network drives each time they start working and disconnect when they finish. The system is also prone to unintentional changes by a user if the user account belongs to a group with access rights, which may cause Windows NT/2000 to stop working.

A better design would be to split the application into two (or more) layers, the server and the UI client, with the server running as a Windows NT/2000 service in a dedicated Windows account and the UI client running in a logged-in user Windows account. In this set up, the server is responsible for controlling the instrument and the UI client is an interface to the server. By separating the the server and UI client in different Windows accounts, the instrument can stay up and running all the time, regardless of when a Windows user logs on or off. Keeping the instrument ready at all times fulfills one of the major requirements in the medical/semiconductor industry.

Figure 1 shows how the overall application setup will look; for simplicity, I have created two Windows NT/2000 local accounts: InstrumentServer, which belongs to the administrators group, and InstrumentServerClient, which belongs to the power user group. Alternately, you can use the built-in SYSTEM account instead of creating an InstrumentServer account.

Figure 1 shows that the server is installed as a Windows service and is running in the InstrumentServer account, communicating with the Instrument over the Serial/USB/Ethernet protocols, while the UI client runs in the logged-in user account InstrumentServerClient. To demonstrate how the overall setup works, I have developed a server (InstrumentServer) IS.EXE as an ATL Windows service that exposes three COM objects and an MFC dialog box-based UI client (InstrumentServerClient) ISC.EXE application. Figure 2 (available online) shows the client user interface.

When you start the client application, the UI will show you the Windows account information for both the client and the server; see Figure 3 (available online).

In this article, I'll review some problems you may encounter when running your application in different Windows accounts. With the help of three COM objects, the server demonstrates these three problems: sharing Windows securable objects across processes, registry manipulation, and network file access.

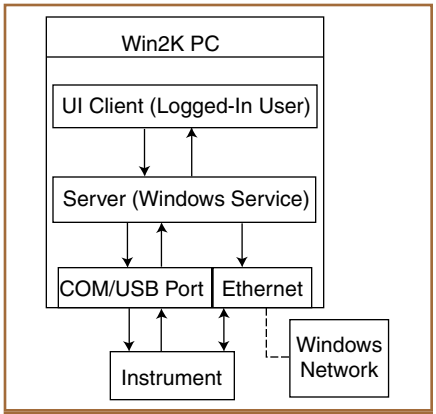


Figure 1 Account setup

Sharing Synchronization Event Objects

The `InformationHandler` object (Figure 4, available online) demonstrates sharing two same-named Windows synchronization `Event` objects with the client. It exposes three methods that clients can use to start computation, get the current counter value, and stop computation. When the client requests that the server start computation, `InformationHandler` creates two Windows events named `InstrumentServerComputationDoneEvent` and `InstrumentServerComputationStartEvent`, and starts a new worker thread. This thread computes a counter and signals the `InstrumentServerComputationDoneEvent` event that triggers a client to pick up a new counter value, while the server waits on `InstrumentServerComputationStartEvent`.

The client also opens two same-named Windows events when the `StartComputing(..)` request returns S_OK and waits on the `InstrumentServerComputationDoneEvent` event. After picking up a new counter value, the client signals an `InstrumentServerComputationStartEvent` event that triggers the server to go ahead and compute again. As this process continues, you will notice in the user interface when events go to signal and nonsignal with a new value a counter. In Figure 5 (available online), the event sequence diagram depicts how the communication between the client and server takes place.

What I have explained is a very generic usage of Windows events, which most programmers use in their applications, but what I am going to show you is something different in the context of a Windows user account. Your PC may be part of a corporate Windows NT/2000 domain, workgroup, or a standalone, and your user account may be part of a different user group (assuming the server is installed in the SYSTEM account) on a given PC. I am going to consider the following use cases to show you the client behavior.

SUSHIL SRIVASTAVA *specializes in process control software development and currently works in Oregon. He has eight years of experience, and his expertise is mainly in Windows NT/2000 system-level programming, object-oriented languages and tools, Visual C++ 6.0, 7.0, MFC, ATL, COM, DCOM, and object-oriented CASE Tools using UML notations, and object-oriented analysis and design.*

windows:: developer MAGAZINE

APPLICATION DEVELOPMENT FROM WINDOWS TO WEB

CD-ROM Version 6.0 On Sale Now!

The *Windows Developer Magazine* CD-ROM gives you all editorial content and source code from **December 1991 through December 2002 issues** along with instant access to all installments of:

- George Frazier's *Tech Tips*
- Jeff Claar's *Bug ++ of the Month*
- Petter Hesselberg's *UI Programming*
- Victor Volkman's *Books in Brief*
- Paul Kimmel's *Frameworks*
- and much more!

Order Online!

www.windevnet.com/wdm/cdrom/

for only $29.95*

Discount Key Code: WDCDROM

www.windevnet.com/wdm/cdrom/

For eleven years *Windows Developer Magazine* has been the independent technical magazine for Windows programmers covering articles ranging from quick technical tips to ready to use solutions for complex programming problems.

* *Windows Developer* CD-ROM is only $29.95 plus shipping and handling ($4 U.S. and Canada, $10 all other countries) for Internet orders only. Extra charges such as multiple copy orders, and delivery outside the U.S. will apply; exact charges will appear when online order is placed. CD-ROM orders placed by phone (800-444-4881) or mail (*Windows Developer* CD-ROM, 1601 West 23rd Street, Ste. 200, Lawrence, KS 66046, USA) are $39.95 plus shipping and handling. Mail orders accepted for U.S. delivery only. Credit cards accepted on Internet and phone orders only.

Figure 7 Client registry operation

Running the server in SYSTEM and the client in one of the following account types: administrator, power users group, users group, or guest account. Even though the client is running in an administrator account, when you start computing (by clicking "Start Computing" from the client UI), the client will fail to open the `InstrumentServerComputionDoneEvent` event. If you check "Use NULL DACL" and start the computation again, it succeeds. Please note: You won't be able to start or stop any service in users and guests group accounts from the Control Panel Services applet.

Running both the server and the client in the administrator account. Let's change the server user account using the following steps:

1. Open Control Panel ->Administrator Tools->Services.
2. Look for the "InstrumentServer" service, double click on it or right-click select "Properties," and make sure the service is not running. Select the "Log On" page, and then select "This account." Specify your user account information and click "OK" to save the settings.
3. Now start the client and try to start computing without checking "Use NULL DACL." Computation starts without any problems.

So how does the "Use NULL DACL" checkbox allow the client to communicate with the server successfully, even in the different user accounts? When you select this option in the client user interface, the client sets the server's property `m_bUseNULLDACLForEvent` to True and, at the time of computation start up phase, the server creates Windows events using a NULL DACL instead of using a default security descriptor in the LPSECURITY_ATTRIBUTES argument. Listing 1 (available online) is a Win32 `CreateEvent(..)` method definition, and the first parameter is a pointer to a security attribute.

When you create any securable object such as an event, mutex, semaphore, and so on, Windows assigns it a security descriptor. The security descriptor identifies the object's owner and contains the Discretionary Access Control List (DACL) and System Access Control List (SACL). I am going to focus on DACL, which identifies who is allowed or denied access to the object. DACL contains a list of Access Control Entries (ACE), and each ACE specifies a set of access rights and contains a Security Identifier (SID) that identifies an individual or group account to whom the rights are allowed or denied. If you set the DACL belonging to an event's security descriptor to NULL, a NULL DACL is created and it grants full access to any user that requests it.

Listing 2 (available online) shows how to create a NULL DACL and use it in the creation of events (error handling is omitted to minimize code lines).

If the client doesn't specify the "Use NULL DACL" option, events are created with default security (Listing 3, available online), which means that Windows assigns a security descriptor to the `Event` object with the owner of the server's user account. Processes running in that user account have access to these events.

You can allow access to a group of users by adding ACEs to the DACL. The `InformationHandler` object uses NULL DACL for events that allow the client running in any Windows account to open and synchronize to the same events.

Registry Manipulation

The `RegistryHandler` (Figure 6, available online) demonstrates the same concept when it comes to creating registry keys and subkeys by using either a default security descriptor or NULL DACL. I think if you use the Windows registry economically, it's the best place to store small amounts of data. I store most of the application settings and user options in the registry, even though there are other choices (each has its own pros and cons). In an application setup, when the server and the client run in a different user account, then we know that the client will be accessing registry entries created by the server—or maybe the other way around. There could be other client type utilities that require access to these registry entries created by the server; in these types of scenarios, registry entries should be created with the proper security descriptor. Consider an example where the server stores a full filename path string C:\DATA\PC_LIST_003.TXT at the HKEY_CURRENT_USER\Software\ZabardastJosh\InstrumentServer registry key section, and this file contains a list of computer names that the server accessed in the last 24 hours. A maintenance utility will be reading this registry key to look for an active filename, and once it retrieves the active filename, the utility will pick up all computer names by reading the file. The `RegistryHandler` object exposes methods that can be used by the client and asks `RegistryHandler` to create a registry key, string value, and set string data by using either the default security descriptor or NULL DACL. In my client user interface, you can ask the server to create registry keys and then ask the client to read those registry keys. If the server is running in the SYSTEM account, registry information for the current user is stored in HKEY_USERS\.DEFAULT, not in HKEY_CURRENT_USER, and you will notice in the client user interface whether registry entries are created in HCU (HKEY_CURRENT_USER) or HU (HKEY_USERS).

I am using the ATL `CRegKey` registry class for the registry operation; Listing 4 (available online) shows the `CRegKey` class `Create` method in the `RegistryHandler` object. The sixth argument is a security attribute, and I use the default security descriptor by specifying NULL or NULL DACL.

Figure 7 shows part of the client user interface for registry operation. There are two sections, one for the server and one for the client. I am creating a registry key, "Logging," with a string value "ActiveFileName" and string data "Today.log." The current application setup is running with the client as a power user and the server as a SYSTEM account. First, I created a registry key by checking "Use NULL DACL" and had the client section try to read it back; the registry key read operation succeeded. If I didn't check "Use NULL DACL," clients failed to read a registry key.

Network File Access

The `FileHandler` object (Figure 8, available online) demonstrates a scenario where the server processes a file at a client's request, located on either a local computer or on the corporate network, and returns the result back to the client. This file could be a database file or any file with data in a format that the server recognizes. The problem in this scenario doesn't relate to a file format or how the server will be able to process a file, but whether the server will be able to access this file. If this file exists on the

local computer, it will not be a problem for the server and the file will be processed successfully, assuming file read access is granted. But consider when a file exists on a network location that the client can access because it is running in a user account that has access rights. If the server is not be running in the same user account, access to that network location will fail.

My intention in this section is to show you how to overcome this problem by having the server set up a network connection on behalf of the client. FileHandler has a C++ class, `CNetworkConnectionHandler`, which maintains a network connections map on behalf of the client user account. This map is created and cleaned for each client's session. Here is an example of how it works: When the client asks the server to process a file by passing the UNC (Universal Naming Convention) string \\Zabardast-Josh_Domain\SW_SSHILPI\Flowers_Collection.mdb, the server tries to connect to the network resource Flowers_Collection.mdb by using its own credential in a Win32 `WnetUseConnection(..)` call. If access is denied, it reports this problem to the client. The client user interface has an "Ask For User Login Information" option. If this option is checked, and when the file process operation is initiated, the client is asked to supply its user account credentials; this information is passed down to the server. Using the supplied user account information, the server sets up a network connection to the intended resource location and processes a file. I have implemented a very simple file-processing operation in my client-server application setup. The client browses for a text file and asks the server to process it by calculating the number of characters in this text file and return the total value. Figure 9 (available online) shows an example of processing a "Readme.txt" file on a local PC.

There are two Win32 APIs that I use to determine whether the file is located on a local PC or on a network, and to set up a network connection. I utilize the `WnetGetUniversalName(..)` API to find out whether a particular file is a local file, is on a mapped network drive, or is on a network but its location is not mapped. Listing 5 (available online) is a code snippet from the `FileHandler` object.

`WNetUseConnection(..)` is used for setting up a network connection—this API can automatically select an unused local device to redirect to the network resource. Listing 6 (available online) is a code snippet from my C++ `CNetworkConnectionHandler` class.

When the server gets started, an instance of the `CNetworkConnectionHandler` class is created, and this class enumerates the current network connect and stores it in its map. It also remembers that these network connects are setup or persisted for the current logged-in user account. When the server sets up a network connect for a user account, it checks whether this network connect was already setup by the server or if the server and the client are both running in the same user account that already has a persisted network connection.

Installation

The IS.EXE server and ISC.EXE client apps and source code are available online. I used Visual Studio 7.0 for my development on Windows 2000. In order to run the apps, you should have either Visual Studio 7.0 or should download run-time DLLs from Microsoft's web site.

- Regvsr32.exe InstrumentServerPS.DLL.
- IS.EXE -service.
- Open Control Panel ->Administrator Tools->Services, look for "InstrumentServer" service, double click on it or right-click select "Properties," select the "Log On" page, and check "Allow service to interact with desktop." When the service starts, you will notice a hand-shaking icon on a system tray (near clock) with Server's Windows account information.
- Run ISC.EXE. w::d

| Download code › **windevnet.com/wdn/code/** |

Avoiding references to nonexistent objects; also, an interesting look at main ()

Temporary Object Bug

I DOUBT ANYONE WHO has programmed in C++ for any length of time would disagree that it can be a tricky language. There are all sorts of subtle traps and gotchas that can lead to what looks like reasonable code failing in nonobvious ways. It's probably not too long, for example, before most novice C and/or C++ programmers run across problems that have to do with object lifetimes. The typical problems arise when using pointers; in particular, when you use a pointer to an object that has already been deleted:

```
MyClass *mc = new MyClass;
// Do some stuff
delete mc;
mc->a = 1;  // Uh oh...mc is no longer valid!
```

The more subtle problems occur with function return values. Suppose function `foo()` returns a reference to an object of type `MyClass`:

```
MyClass &foo()
{
    MyClass mc;
    // Do some things
    return mc;
}
```

This is perfectly legal code that is just asking for trouble. The object `mc` will be destroyed when `foo` exits, but the function is returning a reference to it! The calling code now has a reference to a nonexistent object. If you're lucky, your compiler will give you a warning (Visual C++ 7.0, for example, reports "warning C4172: returning address of local or temporary."), but it's not required to do so.

This is pretty basic stuff, and is the kind of thing every C++ programmer has probably been burned on at least once. Things get a little more complicated with temporary objects, however. What happens if I change the definition of `foo` slightly, so that it returns a copy of `mc`, rather than a reference to it?

```
MyClass foo()
{
    MyClass mc;
    return mc;
}
```

When `foo` returns, it will now create a temporary object that will be assigned into the variable specified by the caller. To see how this works, take a look at the code in Listing 1. The output from both Borland C++ 5.5.1 and VC7 is:

```
Default constructor
Copy constructor
Destructor
Returned from foo
Destructor
```

If you really think about it, this seems like there is an object missing. After all, if you look at the pseudocode, here's what should be happening:

```
In foo: mc is declared, which
    calls the default constructor.
foo returns a temporary copy of mc, which calls the copy constructor
The copy of mc is assigned to rv, which calls the copy constructor.
```

But wait…according to the output, the copy constructor is only called once! It would seem that there are three objects here: `mc` (in `foo`), the temporary object, and `rv`. Why aren't three constructors called? The answer lies in an optimization allowed by the C++ Standard to avoid this very inefficiency. If a temporary return value is immediately assigned into an object, the temporary can be constructed at that object's location in memory. This prevents an unnecessary constructor call, which can save a lot of time if the constructor does a lot of work. (See section 12.2, paragraph 3 of the Standard for more details, if you're so inclined.)

The related case is if `rv` has already been declared. In other words:

```
MyClass rv;
rv = foo();
```

In this case, the temporary return value cannot be constructed into the location of `rv`, since by the time of the call to `foo`, `rv` has already been constructed. Therefore, the temporary return value must be

JEFF CLAAR *is a software engineer at Nemesis in Southern California. He divides his time between programming and trying to keep his infant son from eating everything he finds on the floor. You can submit your bugs to him at wdletter@cmp.com.*

Listing 1
Function returning a temporary object

```
// Demonstrates returning a temporary object.
#include <iostream>

using namespace std;

class MyClass
{
public:
    MyClass(const MyClass &)
    { cout << "Copy constructor\n"; }
    MyClass()
    { cout << "Default constructor\n"; }

    MyClass &operator=(const MyClass &)
    {
        cout << "Assignment operator\n";
        return *this;
    }

    MyClass::~MyClass()
    {
        cout << "Destructor\n";
    }
};

MyClass foo()
{
    MyClass mc;
    // Return a copy of mc.
    return mc;
}

int main()
{
    // This code generates the temporary
    // object directly in the location
    // of retval;
    MyClass rv1 = foo();
    // This code generates a temporary
    // object, which then is copied
    // into rv2 using the assignment
    // operator.
    //MyClass rv2;
    //rv2 = foo();
    cout << "Returned from foo\n";
    return 0;
}
```

created as a separate object, and then assigned into rv. Indeed, if you uncomment the code in Listing 1 that declares these same statements, you get the expected output. (I have added which object each line refers to in parenthesis.)

```
Default constructor (rv2)
Default constructor (mc)
Copy constructor (temporary)
Destructor (mc)
Assignment operator (rv2 = temporary)
Destructor (temporary)
Returned from foo
Destructor (rv2)
```

The important thing to note here is that the temporary object is destroyed at the end of the expression where it was created. In other words, the destructor will be called at the end of:

```
rv = foo();
// Temporary is destroyed here
```

This is all well and good, but what happens in the following case?

```
MyClass &mc = foo();
```

Now, instead of copying the temporary object into a new object, I just assign it to a reference. (Note that this is different from the original example. In the first example, I returned a reference to a local object. In this example, I am assigning a reference to a temporary object returned from a function.) What happens now? When does the temporary get destroyed? If the temporary gets destroyed at the end of the expression, as in the aforementioned example, I would end up with a reference to a nonexistent object. However, the C++ Standard grants an exception in this case: If that temporary object is assigned into a reference, the object will not be destroyed for the lifetime of the reference. In other words, unlike returning a reference to a local variable, binding a reference to a temporary object is perfectly legal. The object will be valid as long as the reference is.

The Bug

I guess I should actually say that the reference will be valid as long as the compiler implements it correctly. Eugene Gershnik sent in the code shown in Listing 2. It has a function, foo, that returns a temporary object of type std::vector<char>. This object is assigned into a reference. When run under VC7 with release options, the program runs with no problems. When run with debug options, however, I get:

```
"The instruction at "0x004121b5"
    referenced memory at "0x00000000".
    The memory could not be "read"."
```

My initial assumption was that the release target was optimizing out the line "int m[80] = {0};", since it really doesn't do anything anyway. I figured that this optimization would eliminate the problem. However, I sent this code off to Microsoft for its comments, and Jeff Peil narrowed it down more than that:

> *The problem is caused by enabling edit-and-continue debugging support. This will be fixed in an upcoming release of Visual C++. You can work around this by compiling without edit-and-continue debugging support (generating debug information is still fine, so just switch to /Zi from /ZI). Alternately, you could work around this by leaving edit-and-continue support enabled and changing the code to avoid binding a reference to a temporary returned like this (for instance, see the following):*

```
int main()
{
    std::vector<char> bar;
    std::swap(bar,foo());
    int m[80]={0};
    std::printf("%d\n", bar[0]);
    return 0;
}
```

—Jeff Peil

Jeff's solution works well, with a minimal performance hit. Swapping the contents of two vectors may at first seem like an expensive operation, but in reality all swap does is swap a few internal variables between the two vectors. There are no buffer copies involved, so the operation will complete in constant time. The biggest difficulty here is determining what the problem is in the first place!

Yet another option to work around the problem is to simply move the "int m[80]={0};" expression above the declaration of bar. Since they are not dependent on each other, it shouldn't matter which is declared first.

Redefining main()

Jonathan Hays sent in this note:

> *I was recently helping edit an online C/C++ test, and came across an interesting issue that I don't fully understand. One of the questions was whether or not the main function can be overloaded. Like all good programmers, I cracked open my copy of the ANSI Standard and looked it up. In section 3.4, it says this about main: "A program must contain a function called* main(). *This function is the designated start of the program. This function is not predefined by the compiler, it cannot be overloaded, and its type is implementation dependent."*
>
> *However, in an effort to be thorough, I tested the code in Listing 3 on both VC6 and Metrowerks Codewarrior.*
>
> *It turns out that both of these compilers compile and run them just fine. Am I misunderstanding the Standard here? Granted, this isn't a very serious issue, but I am more than curious to know if they just aren't Standard conforming.*
>
> *—Jonathan*

At first, Jonathan's code looked like it was indeed a bug. The code redefines main() as:

```
A main(int, long);
```

As Jonathan said, Codewarrior and VC6 compile the program properly. VC7 gives a warning:

```
warning C4326: return type of
    'main' should be 'int or
    void' instead of 'A'
```

Borland C++ (5.5.1), on the other hand, reports that:

```
Error E2120: Cannot call 'main'
    from within the program
    in function WinMain
```

My first thought was that the declaration of `main()` was illegal because it must be defined of the form "int main()" or "int main(int argc, char *argv[])". Closer reading of the Standard, however, mentions this in 3.6.1.1:

It is implementation-defined whether a program in a freestanding environment is required to define a main function.

In other words, for a Win32 application (a freestanding environment), a `main` function is not required. Things seem to get a little unclear in the next paragraph, which states that the `main` function shall not be overloaded. This would seem to indicate that the declaration of `main()` in Listing 3 is illegal. However, `main` isn't really being overloaded here, since it is not required in the first place! Personally, I think the Standard is a little unclear on this case, so I'm not going to say one way or the other whether this is truly a bug in the compiler. I'd be interested to hear your opinions about this. (Of course, the truly simple solution is to never use a function called "main" other than for its intended use.)

Conclusion

Thanks once again to everyone for their bug submissions and comments. In particular, Eugene gets a *Windows Developer* T-shirt for his troubles. If you have found a bug in a popular compiler, be sure to send it to wdletter@cmp.com for possible inclusion in the "Bug++ of the Month" column. Get it down to the smallest possible code, and submit it along with the required compiler version and options to reproduce it. If your bug is used, the *Windows Developer* T-shirt will be yours! **w::d**

| Download code › **windevnet.com/wdn/code/** |

Listing 2 Assigning a reference to a temporary object

```
// Problem with reference bound to temporary
// The function foo returns a temporary object
// of type std::vector<char>, which is then bound
// to a reference of type const std::vector<char>&.
// When the expression "int m[80] = {0};" is
// executed,
// the reference bar no longer seems to be valid,
// and the program will crash in the call to
// printf.
// Removing the line "int m[80] = {0};" eliminates
// the problem.
//
// Compile with VC7, with the "Program Database
// for Edit and Continue" debug option.

#include <vector>
#include <cstdio>

std::vector<char> foo()
{
    std::vector<char> ret(20);
    return ret;
}


int main()
{
    const std::vector<char> &bar = foo();
    int m[80]={0};
    std::printf("%d\n", bar[0]);
    return 0;
}
```

Listing 3 Redefinition of function main()

```
// Possible bug in various compilers.
// This program defines a function
// called main, but compiles successfully
// on Visual C++ 6/7 and Codewarrior.
// It does not compile on Borland C++ 5.5.1,
// but not for the reason you might expect.
#include <windows.h>

class A
{
    public:
        int x;
};

A main(int,long)
{
    A foo;
    foo.x = 17;
    return foo;
}

int APIENTRY WinMain(HINSTANCE,HINSTANCE,LPSTR,int)
{
    A squid = main(1,2);
    return squid.x;
}
```

Please send us your best tricks and hacks—those clever pieces of code to make things work the way they should! You'll receive at least $50 for each tip that we print. Send your submissions to wdletter@cmp.com with the header "Tech Tip submission."

Locking Window Updates

BY MATTHEW WILSON
matthew@synesis.com.au

A CLASSIC PROBLEM WHEN updating windows in lengthy and granular operations is that of window flashing, where changing the contents of a control results in the repeated redrawing of part or all of its visible area. When a large number of such updates occur in a short time, this becomes problematic, and the repeated redrawing appears as a very unappealing visual disturbance.

Two common solutions to this problem are to use the API function `LockWindowUpdate` and the WM_SETREDRAW message.

`LockWindowUpdate` is implemented by swapping out the window's normal device context and replacing it with one whose visible region is empty. When `LockWindowUpdate` is called with NULL (which unlocks the window), the original device context is replaced, and the system invalidates an area within it equivalent (in size and location) to that of the temporary one such that the window will receive a request to redraw the modified area. Hence all drawing is effectively performed in a single operation, and the visual effect is much more seamless. The disadvantage of using `LockWindowUpdate` is that it can only be used for one window at a time, so it cannot be used to simultaneously lock a collection of related controls. The first window to call the function is locked, and all other calls preceding the unlock call fail. (This can be easily shown by uncommenting the artificial lock of the run—button on line 93 of the test application implementation file, wndscope.cpp, which is included in this month's code archive.) Interestingly, if two versions of the application using `LockWindowUpdate` are run simultaneously it appears that whichever process calls the function "owns" the lock, removing the ownership of the previously successful call, which reverts to the unlocked behavior of flashing on the string inserts.

WM_SETREDRAW is a message implemented by various standard common and custom controls, including the listbox, combobox, list view, button, and tab control. It works by clearing and setting the window redraw flag. The only slight disadvantage is that the application must invalidate the window rectangle after sending the reactivating message, and therefore the whole visible window rectangle will be redrawn. Use of WM_SETREDRAW generally supersedes use of `LockWindowUpdate`. Of course, for windows that do not support this message, `LockWindowUpdate` remains the tool of choice.

Presented here are two classes from the WinSTL libraries (http://winstl.org/)—window_update_scope and window_redraw_scope—that provide automated scoping of these two forms of locking. (Abridged implementations of the two classes are shown in Listings 1 and 2. The full implementations are provided in the archive, and are also available online at the WinSTL site.) In the constructors of the classes the lock is set, then reset in the destructors: window_update_scope calls `LockWindowUpdate` in its constructor and, if successful, calls `LockWindowUpdate(NULL)` in its destructor; window_redraw_scope sends WM_SETREDRAW (passing False) in its constructor and sends WM_SETREDRAW (passing True). They both take the handle of the window to lock in their constructors; window_redraw_scope takes a second parameter (defaulted to True) that specifies whether the window is invalidated (via a call to `InvalidateRect`) when unlocked.

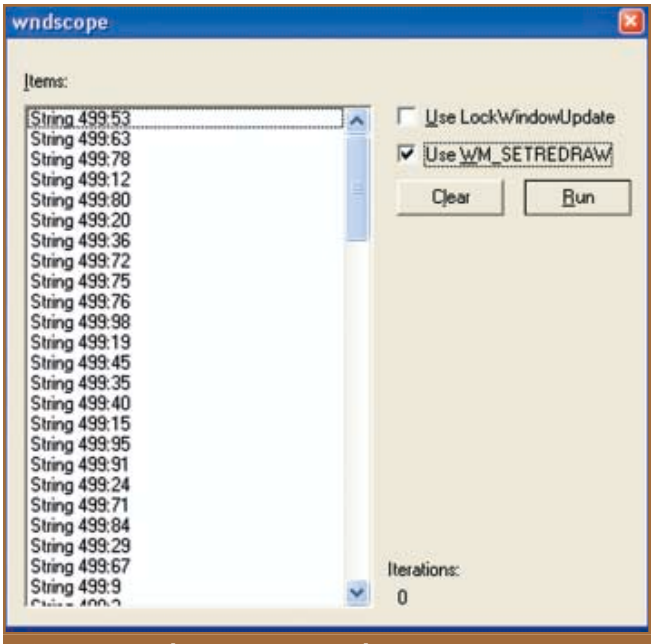


Figure 1 Window scope example

The uses of these two techniques are demonstrated in the accompanying application (Figure 1), which sends many items to a listbox, with or without the two locking techniques just described. To demonstrate the techniques, simply execute the program and select "Run," demonstrating the flashing. Then select, in turn, the "Use LockWindowUpdate" and "Use WM_SETREDRAW" boxes and run, demonstrating the locking.

GEORGE FRAZIER *is a software engineer in the System Design and Verification group at Cadence Design Systems Inc. and has been programming for Windows since 1991. He can be reached at georgefrazier@yahoo.com.*

Listing 1 Abridged version of window_update_scope

```
/* ////////////////////////////////////////////////////////////
 *
 * ...
 *
 * Extract from winstl_window_update_scope.h
 *
 * www:        http://www.synesis.com.au/winstl
 *             http://www.winstl.org/
 *
 * Copyright (C) 2002, Synesis Software Pty Ltd.
 * (Licensed under the Synesis Software Standard Source License:
 *  http://www.synesis.com.au/licenses/ssssl.html)
 *
 * ...
 *
 * ////////////////////////////////////////////////////////// */

...


namespace winstl
{

    class window_update_scope
    {
    // Construction
    public:
        // Takes a HWND and changes it's current update-status,
        // which is set back to the original state in the
        // destructor.
        ws_explicit_k window_update_scope(HWND hwnd)
            : m_bIsLocked(hwnd != NULL && ::LockWindowUpdate(hwnd))
        {}

        ~window_update_scope() winstl_throw_0()
        {
            if(m_bIsLocked)
            {
                // Lock was successful, so undo here
                ::LockWindowUpdate(NULL);
            }
        }

    // Members
    protected:
        ws_bool_t   m_bIsLocked;

    // Not to be implemented
    private:
        window_update_scope(window_update_scope const &rhs);
        window_update_scope const
            &operator =(window_update_scope const &rhs);
    };

} // namespace winstl
```

Listing 2 Abridged version of window_redraw_scope

```
/* ////////////////////////////////////////////////////////////
 *
 * ...
 *
 * Extract from winstl_window_redraw_scope.h
 *
 * www:        http://www.synesis.com.au/winstl
 *             http://www.winstl.org/
 *
 * Copyright (C) 2002, Synesis Software Pty Ltd.
 * (Licensed under the Synesis Software Standard Source License:
 *  http://www.synesis.com.au/licenses/ssssl.html)
 *
 * ...
 *
 * ////////////////////////////////////////////////////////// */

...


namespace winstl
{

    class window_redraw_scope
    {
    // Construction
    public:
        // Takes a HWND and changes it's current redraw-status,
        // which is set back to the original state in the
        // destructor.
        ws_explicit_k window_redraw_scope(HWND hwnd,
                                          ws_bool_t bInvalidateOnUnlock = true)
            : m_hwnd(hwnd)
            , m_bInvalidateOnUnlock(bInvalidateOnUnlock)
        {
            ::SendMessage(m_hwnd, WM_SETREDRAW, false, 0L);
        }

        ~window_redraw_scope() winstl_throw_0()
        {
            ::SendMessage(m_hwnd, WM_SETREDRAW, true, 0L);

            if(m_bInvalidateOnUnlock)
            {
                ::InvalidateRect(m_hwnd, NULL, true);
            }
        }

    // Members
    protected:
        HWND        m_hwnd;
        ws_bool_t   m_bInvalidateOnUnlock;

    // Not to be implemented
    private:
        window_redraw_scope(window_redraw_scope const &rhs);
        window_redraw_scope const
            &operator =(window_redraw_scope const &rhs);
    };

} // namespace winstl
```

Listing 3 Modifying the macro

```
// Original do/while Macro

#define MY_ASSERT_ONE(x) do { \
          if (!(x)) {\
                    _asm int 3 \
          } \
          }  while(0)

// new Macro that removes the loop

#define MY_ASSERT_ONE(x) { \
          if (!(x)) {  \
                    _asm int 3  \
          } \
          }
```

Listing 4 Avoiding a compiler error

```
(a). /* This code will cause a compile error */
void main(void)
{
          int i = 10;
          printf("i: %d\n", i);
          int j = 100;  /* This statement will cause a compiler error since
                           it was not declared at the top of the function. */
          printf("j: %d\n", j);
}

(b). /* whereas this code will not */
void main(void)
{
          int i = 10;
          printf("i: %d\n", i);
          {
                    int j = 100;            /* This statement is fine since it is akin
                                               to defining a local function within the function.  */
                    printf("j: %d\n", j);
          }
}
```

Listing 5 Defining a local function within the function

```
void main(void)
{
          int i = 10;
          printf("i: %d\n", i);
          {
          int i = 100;            /* This statement is fine since it is akin to defining
                                     a local function within the function. */
          printf("i: %d\n", i);
          }
          printf("i: %d\n", i);
}
```

Listing 6 The CRecentFileListEx() function

```
class CRecentFileListEx : public CRecentFileList
{
public:
    CRecentFileListEx(UINT nStart, LPCTSTR lpszSection,
        LPCTSTR lpszEntryFormat, int nSize,
        int nMaxDispLen = AFX_ABBREV_FILENAME_LEN):
        CRecentFileList(nStart, lpszSection, lpszEntryFormat, nSize,
nMaxDispLen)
    {
    }

    BOOL GetDisplayName(CString& strName, int nIndex,
        LPCTSTR lpszCurDir, int nCurDir, BOOL bAtLeastName = TRUE) const;
};

BOOL CRecentFileListDlgApp::InitInstance()
{
    ...

    LoadStdProfileSettings();  // Load standard INI file options (including
MRU)

    // Delete the m_pRecentFileList created in the LoadStdProfileSettings.
    delete m_pRecentFileList;

    // The nSize argument of the constructor is set to four because the
    // LoadStdProfileSettings takes a default of four.  If you specify a
    // different value for the nMaxMRU argument you need to change the
    // nSize argument for the constructor call.
    m_pRecentFileList = new CRecentFileListEx(0, FileSection, FileEntry, 4);
    m_pRecentFileList->ReadList();

    ...
}
```

A Better Macro Wrapper for Visual C++

BY JOHN SZAKMEISTER
jszakmeister@actelesys.com

THE FEBRUARY 2003 "TECH TIPS" column included a tip on using the do/while construct to avoid some pitfalls in defining some macros ("do/while Macros for C++" by Raja Venkataraman). This tip is useful, but I thought I'd offer my two cents to improve upon it. There is actually an easier way to get the same result. In C and C++, you can insert curly braces anywhere inside a function to section off a piece of code. I've found this ability to be extremely helpful since I do both Windows and embedded programming. Most cross-compilers do exactly what you tell them (even though some optimizations may be performed), so wrapping something with a do/while loop can actually result in code being generated for the loop. A simple way to avoid this is not to wrap it with a do-while loop, but just wrap it in braces. I also use this feature to help inline several slightly more complicated functions in C, but get to keep the speed that I desperately need to process other things.

Listing 3 shows how the original Tech Tip code would change. The wonderful (and harmful if not used correctly) feature of this technique is that you can also define local variables that will not interfere with other variables in your function. For instance, you normally can't declare a local variable after the first code statement in a .c file (Listing 4(a)). Listing 4(b), however, shows a code segment that works fine because of the new macro. Here you could even change the variable j to i and it would leave the outer variable untouched. So the code in Listing 5 would produce the following output:

```
i: 10
i: 100
i: 10
```

Displaying Full Path Names in the Recent File List Menu Item

BY PABLO PRESEDO
ppresedo@hotmail.com

RECENTLY, SOMEONE SENT ME an e-mail asking me how to get an MFC application to display the file's full path and name in the recent file list. The default behavior is to display the full path only if the current directory is different than the directory where the file is located. It will also abbreviate the path name if it is too long.

The filenames that are displayed in the recent file list are created by the `CRecentFileList::GetDisplayName` function. This function is called by `CRecentFileList::UpdateMenu`, which is called by the `CWinApp::On- UpdateRecentFileMenu` function. The `CWinApp::OnUpdateRecentFileMenu` function is called in response to the ON_UPDATE_COMMAND_UI(ID_FILE_MRU_FILE1, `OnUpdateRecentFileMenu`) message map entry found in appcore.cpp.

```
void CWinApp::OnUpdateRecentFileMenu(CCmdUI*
  pCmdUI)
{
    ASSERT_VALID(this);
    if (m_pRecentFileList == NULL)
    // no MRU files
        pCmdUI->Enable(FALSE);
    else
        // *** This function will
        //call GetDisplayName. ***
        m_pRecentFileList->UpdateMenu(pCmdUI);
}
```

One solution to this problem is to delete the `CRecentFileList* m_pRecentFileList` after the `LoadStdProfileSettings` call made in your `CWinApp` derived `InitInstance`. We replace this with a class of our own that was derived from `CRecentFileList` (Listing 6). Our `CRecentFileList` will override the `GetDisplayName` function.

Make sure the `nSize` argument of the `CRecentFileList` derived class constructor is set to the value of the `nMaxMRU` argument of the `LoadStdProfileSettings`. That's all that there is to it. In this month's code archive, I have included an example that demonstrates this, as well as how to get an MFC dialog application to display a recent file list. By default, MFC dialog applications do not support a recent file list. **w::d**

| Download code › **windevnet.com/wdn/code/** |

Reconciling a changes-only copy of your data can be tricky, particularly if it contains new records as well as updates

Reconcile Your Data with a Smarter Merge

THE DISCONNECTED NATURE OF ADO.NET brings up a number of new issues. Data conflicts, record locking, and optimistic concurrency are only the most featured ones. Another, more subtle, problem can easily show up if you're building real-world, high-scalable applications and make intensive use of disconnected data. At the foundation of the ADO.NET disconnected model lies the assumption that you load and save data at different times with no form of client/server connectivity existing in the mean time. How do you deal with server-generated values? And do you merge and reconcile the client copy of the data with the modified copy produced by the database server? And, more importantly, how do you accomplish that using ADO.NET tools?

In general, when it comes to submitting a bunch of updates to a remote database, the right tool to use is batch update. ADO.NET, in particular, provides a handful of interesting features including the capability of retrieving and posting back server-side generated values. Turned on by default, this feature works only if you use stored procedures or SQL batches to perform updates. The reason is that an extra SELECT statement must be added to the original command for ADO.NET. Once the command terminates, the ADO.NET infrastructure retrieves those values and automatically updates the client copy of the DataSet. I discussed underpinnings and implications of this technique in the May "Inside .NET" column.

In this article, we'll use the same technique in a particular scenario; specifically, when you have a separate data tier that receives a changes-only DataSet from the client.

Where the Problem Lies

High-scalable, real-world applications have a data tier placed somewhere in between the client and the remote database. Moving data from the client to the database and vice versa is an operation that can affect the throughput of the application and its overall performance. The DataSet is a serializable object and, as such, can be transferred from tier to tier. For performance and scalability reasons, though, you normally want to avoid transferring the whole set of data that the client holds and displays. Well-designed clients extract a subset of rows from the DataSet tables and send only them. It goes without saying that the rows extracted are those that have been changed since the last commit to the database.

Suppose now that a user has added new records to one of those tables. Suppose also that one of those tables has an autoincrement column. ADO.NET can generate autoincrement values, but those are fake numbers that are guaranteed to be unique only in the context of the DataSet. When you submit a new row to the database, the batch update process intelligently recognizes autoincrement and read-only columns and excludes them from the commands. In this way, the ADO.NET-generated increment value is never passed to the server, and at the end of the batch it's automatically replaced with the actual, server-side generated value. So far, so good. What happens, though, if the DataSet being sent to the data tier for update is a changes-only copy? The server-side values are replicated in the copy of the DataSet, not the original.

This is not too difficult because the DataSet features the `Merge` method, which is used to merge two DataSet objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a database into an existing DataSet. This allows the client application to have a refreshed DataSet with the latest data from the data source. When you call `Merge` to reconcile the original copy of the DataSet with the changes-only copy modified on the server, something happens that is not exactly what you want. The `Merge` method has no way to distinguish between two otherwise identical rows that now have different values in the autoincrement field. The row in the source DataSet retains the ADO.NET-generated value; the copy of the row that was passed with the changes-only DataSet is now updated with the real value for the column. For the `Merge` method, they are actually distinct rows and both are left in the DataSet at the end of the merge.

We'll first review the basics of the `GetChanges` and `Merge` methods and then discuss a workaround for the aforementioned problem.

Getting Changes

The DataSet object provides an ad hoc method to let you obtain a child DataSet made of changed rows only.

```
DataSet changes = sourceDataSet.GetChanges();
```

The `GetChanges` method walks through the collection of tables in the source DataSet and copies all the rows that result changed. The

DINO ESPOSITO *is Wintellect's ADO.NET and XML expert and is a trainer and consultant based in Rome, Italy. He is a contributing editor to* MSDN Magazine, *writing the "Cutting Edge" column, and is the author of several books for Microsoft Press, including* Building Web Solutions with ASP.NET *and* Applied XML Programming for .NET. *Contact him at dinoe@wintellect.com.*

method has an overload that allows you to select rows based on the type of change. For example, the following code snippet shows how to select only the rows that have been added.

```
DataSet added;
added =
sourceDataSet.GetChanges(DataRowState.Added);
```

GetChanges works on all tables contained in the DataSet. There's no direct way to select only the changes occurred on a particular table. In this case, you have to use the GetChanges method defined in the DataTable class.

Note that the changes-only DataSet contains all the tables defined in the schema regardless of whether they have been changed or not. If a table results unchanged, it will be empty in the final DataSet. The GetChanges method works by first cloning the schema of the original DataSet—which ensures that all the tables are there—and then iterating through the tables looking for changes. To clone the schema of a DataSet, you use the Clone method. No data is copied in this case. To obtain a deep copy, in which both schema and data are duplicated, you must resort to the Copy method.

During the copy, constraints are suspended on the DataSet being created. The EnforceConstraints property is temporarily set to False. When the method terminates, constraints are set as in the original. Finally, note that the GetChanges method returns NULL if the original DataSet has no pending changes. The HasChanges method is used to check for this condition.

The Merge Operation

The Merge method is typically called at the end of a series of procedures that validate changes, reconcile errors, and update the data source with the changes. The Merge operation is strictly connected with the update process and the GetChanges method in particular. It's by using the Merge method that you finally refresh the existing DataSet with the version of the DataSet resulting from the data tier.

The Merge method compares and manages rows as well as columns. The first operation that Merge accomplishes is the comparison between the schemas of the involved DataSet objects. There might be situations in which the data and the middle tier also modify the schema of the original table, typically adding columns.

If the DataSet being merged with the original contains extra columns, what happens depends on the value assigned to the MissingSchemaAction argument. The Merge method features several overloads, as shown here:

```
public void Merge(DataRow[]);
public void Merge(DataSet);
public void Merge(DataTable);
public void Merge(DataSet, bool);
public void Merge(DataRow[], bool,
  MissingSchemaAction);
public void Merge(DataSet, bool,
  MissingSchemaAction);
public void Merge(DataTable, bool,
  MissingSchemaAction);
```

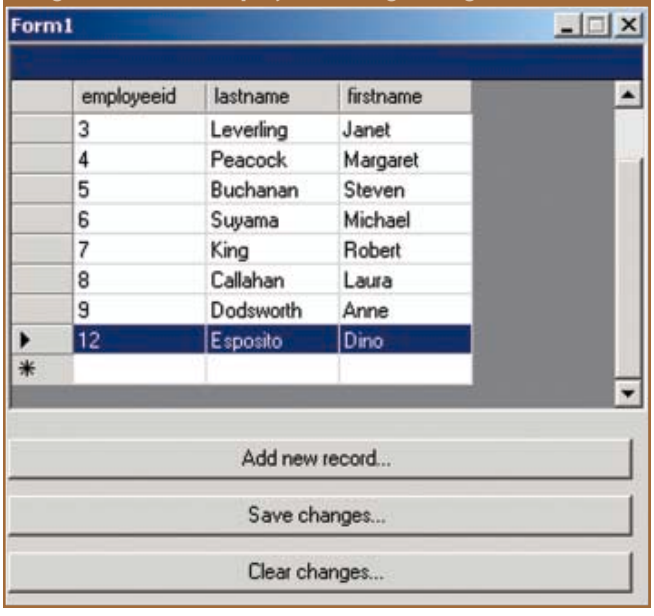
If missing, the MissingSchemaAction is set to Add, meaning that the merged DataSet will contain the added schema and data. Other options are AddWithKey, Error, and Ignore. The Boolean value in the signatures indicates whether the changes made to the original DataSet should be maintained. The default is False, meaning that the state of the rows is accepted and changes that are saved to the database committed.

After merging schemas, the data is merged, and after that, constraints, relations, and extended properties are merged. During a merge, constraints are disabled. At the end of a merge, should any constraints be impossible to enable, a ConstraintException is thrown. The merge has no effect until the constraints are correctly restored. The EnforceConstraints property is automatically set to False; all invalid rows

are marked in error. Errors must be cleared before the `EnforceConstraints` property can be set back to True.

When merging a new source DataSet into the target, rows are matched using primary key values. When the primary key column is also an autoincrement column, you run into a snag—the same row is represented in the two DataSet objects to merge with different values. Both values are, for different reasons, unique and valid in their own context. Only one of them, though, is acceptable. We need to get rid of the original value and replace it with the more-reliable, server-generated value in the changes-only DataSet.

Figure 1 The user has added a new record to the original DataSet displayed through the grid



The Sample Application

In this code snippet, you can see a piece of code that runs when users click the Save button on the sample application of Figure 1:

```
private void button2_Click(object sender,
  System.EventArgs e)
{
    Merger m = new Merger();
    DataSet ds = m_dataSet.GetChanges();
    ds = m.Save(ds, "Employees");
m_dataSet.Merge(ds);

    // bind to grid
    BindData();
}
```

The `Merger` class represents a middle-tier component that takes care of connecting to the data source and performs reads and updates. You pass the subset of the current DataSet that only contains changes and get back a new DataSet. If no row value is modified on the server, then the returned DataSet is entirely contained in the original as it was at the beginning of the procedure. In case of autoincrement columns and insertions, there's a significant difference between the two—the value of the autoincrement column. The `Merge` method in the code snippets shown here loads any changes that occurred on the server in the original DataSet. Or at least this is what you expect will happen.

The employeeID column is an autoincrement column. The ID for the last record has been automatically generated by ADO.NET. That value will unlikely be the same that the database will assign for the same field. However, if you simply perform the merge as in the aforementioned code, what you get is exactly what Figure 2 shows.

Figure 2 The last record is added twice; the second insertion is due to the merge process that fails to recognize the record

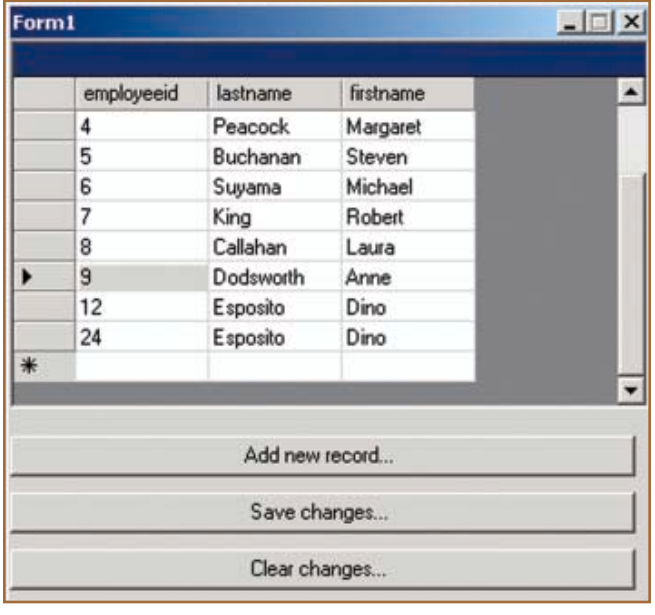
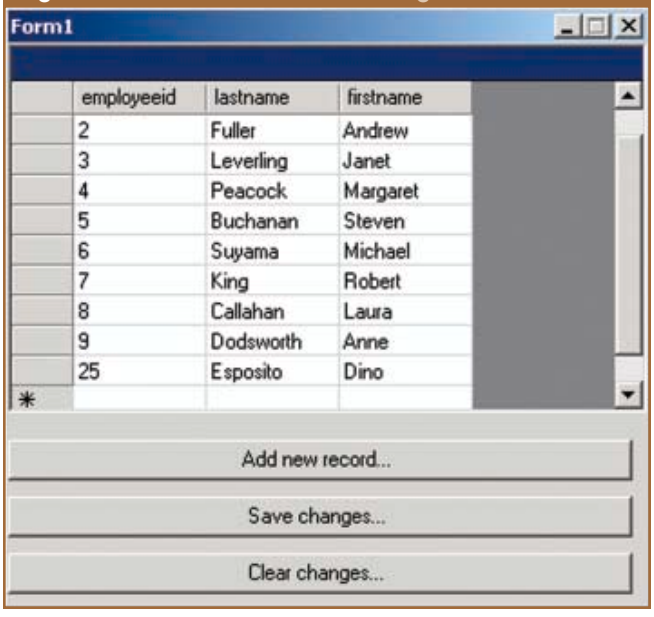


Figure 3 The final result of the merge



The true autoincrement value is 24. The other record is the old copy in which the autoincrement value has been generated by ADO.NET. Since the key values are different, the framework can't distinguish between the two. As a result, a new undesired record is added.

Smart Merging

Smart merging is a procedure that replaces the undesired records with others that have been modified on the server. Of course, it's worth noting that such a problem doesn't occur in all cases, but only if the autoincrement column is also the table's primary key. All in all, this is a rather common scenario.

The following code illustrates a `Merge` method defined as a middle-tier object. The method takes two DataSet objects and merges them. In this particular implementation, it also takes a third argument, being the name of the single table to process. A more general implementation would loop through all the tables and also consider other aspects of merging.

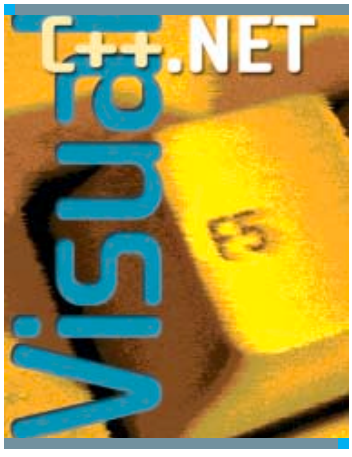
```
public DataSet Merge(DataSet source,
    DataSet changes, string member)
{
    // First, remove added
    // records from the source
    DataTable dt = source.Tables[member];
    DataRow[] rows = dt.Select(
        "", "", DataViewRowState.Added);
    foreach (DataRow row in rows)
        dt.Rows.Remove(row);

    // Second, merge as usual
    source.Merge(changes)
    return source;
}
```

The idea is that you first remove from the original DataSet all the records that have been added. Next, you merge as usual. In this way, you basically partition the original DataSet in two disjoint segments. This prevents at the root any possible data duplication. Figure 3 shows the final result. The source code for the sample app is available online. w::d

| Download code › **windevnet.com/wdn/code/** |

Considering security and stability in an app that has multiple application domains

App Domains and Threads



WELCOME TO THE FIRST VC++.NET Expert column. Every other month, I will be digging deeper into .NET, and taking a look at how .NET works and how to leverage it to your advantage. Of course, to really get the most out of .NET you need a powerful language, and the best candidate is Visual C++.NET. The Managed Extensions for Visual C++.NET 2003 gives you access to all of the features of .NET that are available to other languages and, in addition, it also allows you to integrate native code into your .NET code. This is a facility that is not available to other languages. For the power programmer, there is no choice other than to use Visual C++.NET.

In this month's column, I will address one of the most basic features of .NET: executing code in an application domain. An application domain is a unit of isolation in a .NET process; each process will have at least one application domain, which will have the name of the process. Every Win32 developer will be aware that code in a process is executed on an operating system thread and that a process can have more than one thread. .NET also has a concept of threads (which are currently based on operating system threads). This raises the question of how .NET threads and application domains are related.

Most .NET developers are not concerned with application domains because most .NET processes will run with a single domain. Developers who write ASP.NET code, or assemblies that will be hosted in Internet Explorer, should be concerned with application domains because the nature of their work is that they never write the process in which their code will run. Instead, these assemblies are loaded and executed in a process provided by the system (the ASP.NET worker process or IE6), which could host code from another source. .NET provides isolation in the form of application domains so that code in one hosted application cannot affect the code in another application.

By "affect" I mean either deliberately, by accessing objects directly in another application, or by accident when an exception is thrown and unhandled. Application domain isolation means that all communication between objects in different application domains involves .NET remoting and hence context boundaries are crossed through the help of message sinks. Of course, this assumes that such code has the appropriate evidence to satisfy code access security that such cross domain calls are allowed.

If an exception in an application is not handled, it will kill the application, but domain isolation means that other applications in the same process (but in different domains) are unaffected and will continue to run. A side effect of this isolation is that you can debug an application without affecting the other applications in the same process or killing the process itself. XP has the `DebugActiveProcessStop` API to detach a debugger from a process without killing it but, of course, it does mean that each application is isolated in its own process, and processes are expensive resources.

Hosting Assemblies

Since an application domain is used to host a .NET application within a .NET process, it makes sense for an application domain to be the location where information about an application is stored. A cursory glance through the members of the `System::AppDomain` class gives a good indication of some of the other responsibilities of domains. The application domain will host all the assemblies used by an application and these are accessed through `AppDomain::GetAssemblies()`. If you decide to host the run time in your own process, you can choose to indicate that assemblies are loaded as "domain neutral," which means that when assemblies are loaded, they are available to all domains rather than to a specific domain. To do this you have to explicitly start the run time using native code, and for the purpose of this discussion, I will ignore this situation.

Of course, the Fusion assembly locator technology will need to locate the private assemblies requested by an application (through the `Load()` or a member of the family of `CreateInstance()` methods to load a type from a specified assembly) and the directory that is the base of this search is held in the `BaseDirectory` property. This property only has a `get` method, so once the application domain has been created you cannot change this value. However, you can append to the path used by Fusion to find private assemblies by changing the `RelativeSearchPath` property. Again, this property is read only, but you can change it by calling the `ClearPrivatePath()` to clear the search path and call `AppendPrivatePath()` for each subfolder that you want to add to the path. Changing the search path gives you more flexibility. You can only specify this value before any attempt to locate assemblies has been performed by the domain, hence before any code is executed. To change this value for the default domain (the first domain created when the process is created, which has the name of the process), you should use the <probing> entry in the application's configuration file.

By default, the application's configuration file has the name <application name>.exe.config, where <application name> is the name

RICHARD GRIMES *speaks at conferences and writes extensively on .NET, COM, and COM+. He is the author of* Programming with Managed Extensions for Microsoft Visual C++.NET *(Microsoft Press, 2003). If you have comments about this topic, Richard can be reached at richard@richardgrimes.com.*

of the application process. However, the AppDomain class has a read-only property called SetupInformation, which is an AppDomainSetup object that has a property with the name of the configuration file. Yet again, once the domain has started executing code, you cannot change this value, but if you create a new domain you can specify a file other than the default configuration file:

```
AppDomainSetup* ads = new AppDomainSetup;
ads->ConfigurationFile = S"data.config";

// Create the application domain with
// the same evidence as the current
// domain, and get its configuration
// from data.config
AppDomain* ad;
ad = AppDomain::CreateDomain(
   S"new domain",
   AppDomain::CurrentDomain->Evidence,
   ads);
```

Running Threads

All code in an application domain will be executed by a thread. A thread has an attached security principal object that identifies the thread for role-based security. This means that code can behave in one of several ways according to the thread principal, and since this is a behavior of how code is executed, the principal is a member of the Thread class. Each thread running in an application domain can have a different principal, but if no principal is set then the default value is obtained through the application domain and is set by calling AppDomain::SetThreadPrincipal(). Furthermore, the code that a thread will execute may behave in a culture-specific manner. The Thread class has two properties, CurrentCulture and CurrentUICulture, which identify the culture of the thread. Library code can read these properties on the current thread and make adjustments according to the culture. The difference between the two is that CurrentUICulture is specifically used by the ResourceManager class to load culture-specific resources, whereas CurrentCulture is the general-purpose property used by classes, such as the string formatter.

A thread will run in an application domain, and you can get the domain from the thread through Thread::GetDomain(). But the opposite is not true: There is no method on the domain to get the collection of threads that is running in the domain. The nearest you can get to this is the Process::Threads property; however, be aware that this will return a collection of ProcessThread objects—operating system threads—and not all of these will be a thread running .NET code. There is no documented way to obtain a Thread from a ProcessThread. Clearly, the implication is that a .NET thread is not constrained to a specific application domain.

When your application process is started, a single thread is created to run the entry point code in the default application domain; this is your main thread. The process stays alive as long as the main thread stays alive. Threads can be background or foreground; a foreground thread keeps the process alive whereas a background thread will be killed by the application when the process dies. The main thread is created as a foreground thread and, although you can change it to a background thread, it has no affect on the lifetime of the process. In your applications it is more important to keep the main thread alive than any other thread. An unhandled exception will kill the executing thread, so it is vital that you handle all exceptions in the main thread because if the main thread dies, your process will also die.

Threads usually execute code in just one domain, but if a thread has access to another domain, then it can call objects in that domain. Typically, such access occurs because a thread has created the other domains through a call to AppDomain::CreateDomain(). There is no mechanism in the framework class library for a .NET thread to enumerate the domains running in the process. However, Microsoft does provide a COM object to do this called CorRuntimeHost, which implements the ICorRuntimeHost interface. This interface has a method called EnumDomains() that returns an enumeration handle, and it also has a method called NextDomain(), which is used to iterate through the items in the enumeration identified by the handle. NextDomain() returns an IUnknown pointer, which is actually a System::AppDomain object:

```
// Compile this as managed code
// hEnum is the enumerator handle
IUnknown* pAppDomain;
while (S_OK ==
   cor->NextDomain(hEnum, &pAppDomain))
{
   Object __gc* o;
   IntPtr i(pAppDomain);
   // get the .NET object for this
   // COM interface
   o = Marshal::GetObjectForIUnknown(i);
   AppDomain __gc* pApp;
   pApp = dynamic_cast<AppDomain*>(o);
   Console::WriteLine(
      pApp->FriendlyName);
   pAppDomain->Release();
}
```

The thread running this code has access to the other domains in the application if code access security allows it. The thread can then call AppDomain::CreateInstance() and create an object in the other domain. However, since the access to the object will be across context boundaries, the class must be serializable or marshal by reference or else an exception will be thrown. If it is serializable, then the access across the domain boundaries will result in a copy being created and called in the caller's domain. If the object is marshal by reference, then the object code will be executed in the domain where it has been created. In this latter case, the thread will run code in one domain (the code that initiates the call), then it will run code in another domain (when the object's code is executed).

One reason to have an additional domain is to run code that you know might throw an exception that you do not want to catch, but you also do not want to kill the process. In this situation, it is imperative that this code does not run on the main thread because this could kill the process.

There is another situation when a thread could execute code in more than one domain. Each process has a single thread pool. When you make a request from any application domain in the process to run a delegate on a ThreadPool thread, a thread will be selected from the process-wide thread pool, and once this thread has done its work, it will be returned to the pool where it can be used by any code in any application domain in your process. If the delegate code throws an exception, the thread will be killed but the exception will be suppressed by the run time. Dangerous code like this, run on a ThreadPool thread, will neither kill the domain, nor the process.

Native Code

Finally, it is worth pointing out that when a thread runs native code through Platform Invoke, Com Interop, or via C++ IJW (It Just Works!), the thread will execute outside of an application domain. Indeed, this caused a problem with Version 1.0 of the run time where a thread calling into managed code would always run in the default domain because the run time omitted to save the original domain before the native code was called. This problem has been fixed with Version 1.1 of the run time (Everett).

.NET provides application domains to keep your code isolated. Code is executed by a thread and, although threads typically call code in a single domain, they are not constrained to do so by the run time. Threads can execute code in more than one domain, and can execute code outside of any domain. If you write an application that has multiple application domains, it is a good idea to execute the code in the domains on threads that are specific to those domains or on a ThreadPool thread. The first thread in the process defines the lifetime of the process, so always ensure that you do not execute code that is likely to throw exceptions on this thread, or at least ensure that you have adequate exception handling for this thread. w::d

| Download code › **windevnet.com/wdn/code/** |

ADVERTISER INDEX

GET ADDITIONAL INFORMATION ABOUT PRODUCTS AND SERVICES YOU SEE ADVERTISED FAST!

PHONE: Contact the vendor directly using the information in the advertisement.

WEB: Go to the development tool page on our web site, www.windevnet.com. From there you can link to the advertisers below.

www.windevnet.com

windows:: developer NETWORK
APPLICATION DEVELOPMENT FROM WINDOWS TO WEB

READERS' FORUM

Send Letters to wdletter@cmp.com. Letters may be edited for style and length.

Network-Aware Apps Are Phat

To *Windows Developer*,
[Regarding "Survival of the Fattest," by Petter Hesselberg, December 2002] Great reading! My sentiments exactly. The browser web-application stuff reminds me of the story, "The Emperor's New Clothes." Everybody keeps saying it's great. God forbid you start questioning why a new application should be a native application (either Win32 or Mac OS X) instead of the complex web thing.

The only part of the article I didn't agree with is the title. Not that a "fat app" is a bad thing, just that the distribution could be painful (running around with CDs, and so on). New app development should take advantage of bandwidth in the network/intranet/internet to deliver new functionality when it's available. I would not call these apps "fat apps" but rather "smart apps" or "network-aware apps." Part of the functionality of the "smart app" is to always be checking for updates to its components. If a new version of a component exists, deliver and replace it, either automatically, or with user intervention. Microsoft uses this idea to keep Windows updated. An entirely new application can be delivered this way. E-mail the URL where the app is located and away you go. Keep up the good work!

Thanks,
Jonathan Ott

Submit new product announcements to wdletter@cmp.com.

TALtech Announces Bar Code ActiveX LL Control

TALtech has announced its Bar Code ActiveX Control for MS Access, Excel, and Word. This new "limited license" version is based on TALtech's Bar Code ActiveX PLUS version but is limited for use on up to five workstations instead of unlimited distribution. Bar Code ActiveX Control allows users to print high-quality bar codes on labels or documents from their Windows applications. It supports data binding, which allows users to incorporate bar coding into database reporting and labeling applications. The new Bar Code ActiveX LL is $395.00. Visit the company's web site for further details.

TAL Technologies
800.722.6004
www.taltech.com

Datawatch Delivers VorteXML Server

Datawatch has delivered VorteXML Server, which enables users to automate the process of transforming legacy data into valid XML. The product complements VorteXML Designer, which allows users to visually extract, transform, and map data from structured text output into valid XML without programming. VorteXML Server offers users the ability to convert high volumes of text data to XML; automate complex conversions and transformations; invoke conversion remotely through a web service via Java, .NET, or any other SOAP-enabled client; run conversions on a recurring basis; and trigger XML conversions based on file creation.

VorteXML Server is priced at $7900 per server (up to two CPUs—a VorteXML license for each additional two CPUs is $1999). Service Bureau and ASP pricing is available by request.

Datawatch Corp.
978.441.2200
www.datawatch.com

MetaProducts Ships Web Studio v4.0

MetaProducts has shipped Web Studio v4.0, a multiuser Windows application that allows users to develop and support web sites. Web Studio includes a text editor with development tools for working with HTML/XHTML, JavaScript, VB Script, Cascading Style Sheets (CSS), Perl, PHP, and other web technologies. Web Studio features a customizable user interface, spell checking, HTML code validation, autotext and autocompletion of commonly used strings, and the ability to insert any tags from the main menu, toolbars, or hot keys. HTML pages, CSS rules, and JavaScript functions are generated automatically. The program's FTP client supports mirrors and multiple servers.

Web Studio v4.0 costs $79.95 and can be purchased online at the company's web site. Multiuser and site licenses are available.

MetaProducts Corp.
www.metaproducts.com

Telelogic Releases DOORS 6.1 and Tau Generation 2.1

Telelogic has released a new integration between two of its product families: Telelogic DOORS 6.1 for requirements management and Telelogic Tau Generation 2.1 for visual development. This bidirectional integration provides a role-based interface that allows users to bridge the gap between requirements and standards-based systems and software models for greater lifecycle accountability, automatic error checking, and early prototyping. The Telelogic Tau Generation 2.1-DOORS 6.0 SR1 integration is currently available on Windows 2000 and XP platforms.

Telelogic
877.275.4777
www.telelogic.com

Graphics Server Technologies Ships Graphics Server .NET

Graphics Server Technologies has shipped Graphics Server .NET, a new component that provides 100 percent managed code support for Microsoft's .NET development environment. Graphics Server .NET provides a collection of core business and scientific graphing features including: a wide variety of graph and chart types; easy-to-use property pages to customize graphics; statistical overlays; limit lines; multiple data sources in graph or even a series; Resource Center with extensive Visual Basic and C# code examples; and extensive documentation.

Graphics Server .NET is priced at $899.00 per developer and requires no additional server licenses for applications deployed on the Web. See the company's web site for more information.

Graphics Server Technologies
206.625.6900
www.graphicsserver.com

AidAim Software Announces EasyTable 4

AidAim Software has announced EasyTable 4, a BDE alternative database system providing access to a database in its own single-file format. Version 4 adds SQL'92 support. The implemented SQL subset includes data selection and data modification commands as well as changing data scheme operators. EasyTable does not require BDE (but is compatible with it) or any additional DLLs; it compiles directly into your executable file. EasyTable has additional functionalities such as BLOB data compression, strong encryption of both table and whole database, multi-indexes, table restructuring and

NEW PRODUCTS

repairing, shareable in-memory capabilities, in-memory mode, and more. Visit the company's web site for further details.

AidAim Software
www.aidaim.com

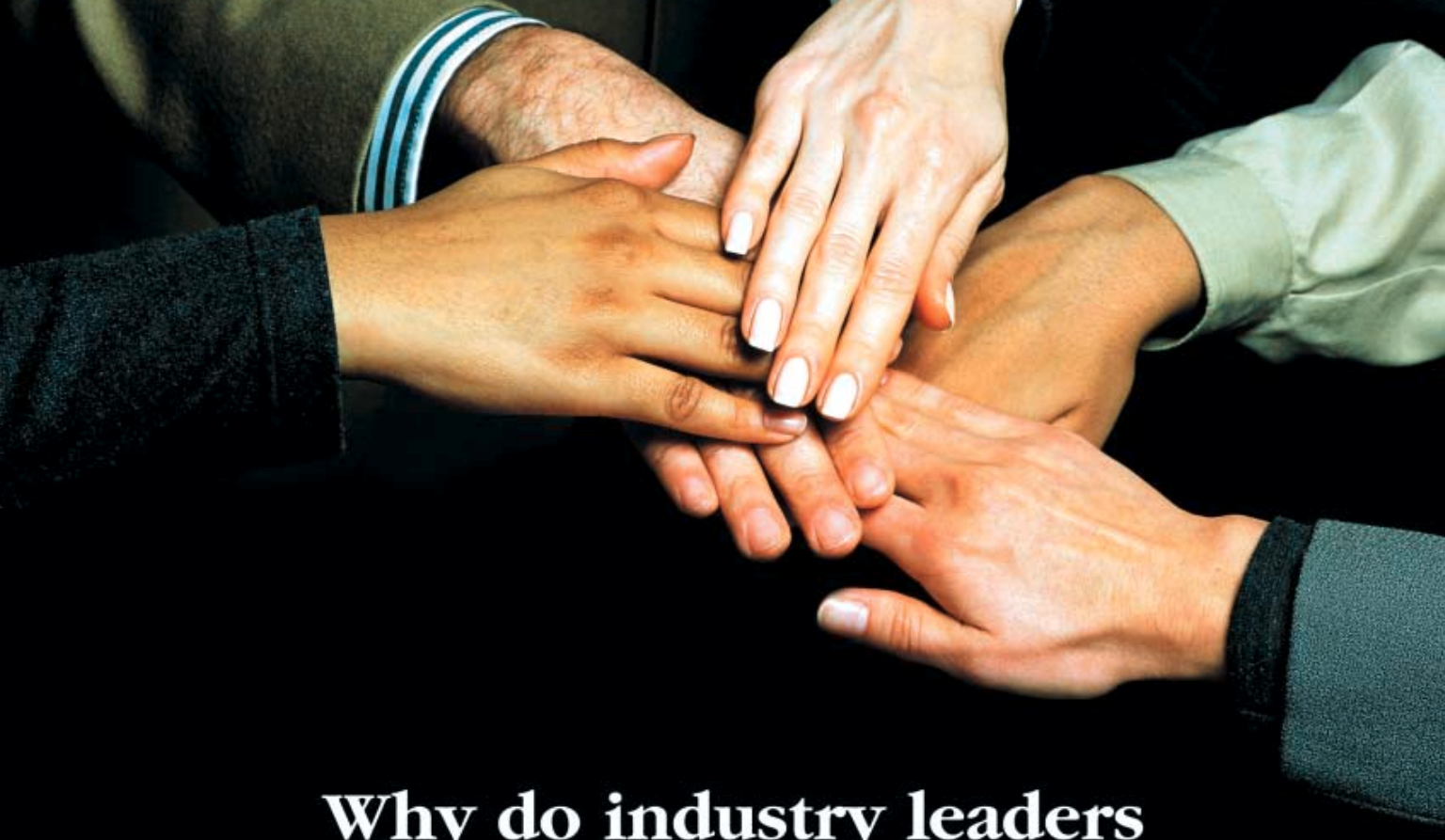
ExclamationSoft Introduces SMTP.NET 2.0 and EmailQ.NET 1.0

ExclamationSoft has introduced SMTP.NET 2.0 and EmailQ.NET 1.0. Both products offer developers extensive e-mail functionality with shortened development and testing cycles. SMTP.NET extends the basic functionality found in .NET e-mail libraries, and includes features such as authentication and the ability to specify multiple mail servers for failover and redundancy. Other features include built-in database functionality such as mail merge and personalization, HTML and plain text, and rich messaging formatting.

EmailQ.NET 1.0 is an e-mail queuing component that integrates with SMTP.NET to schedule e-mail delivery, off-load e-mail transmissions, and track sending progress. Prices start at $99.95 for SMTP.NET 2.0 and $149.95 for EmailQ.NET.

ExclamationSoft Corp.
866.489.0111
www.exclamationsoft.com

DEVELOPERS' MARKETPLACE



Why do industry leaders choose Zero G?

Developers already know that Zero G's InstallAnywhere® and PowerUpdate® are critical for their software installation and updating needs. But, buying from us means you get more than just our award-winning multi-platform software deployment solutions. You also get our customer service, our world-class technical support, and best of all, our people – a team of skilled professionals who are committed to the partnership that starts the minute you download one of our products. That's why industry leaders like Sun Microsystems, Novell and Borland choose us.

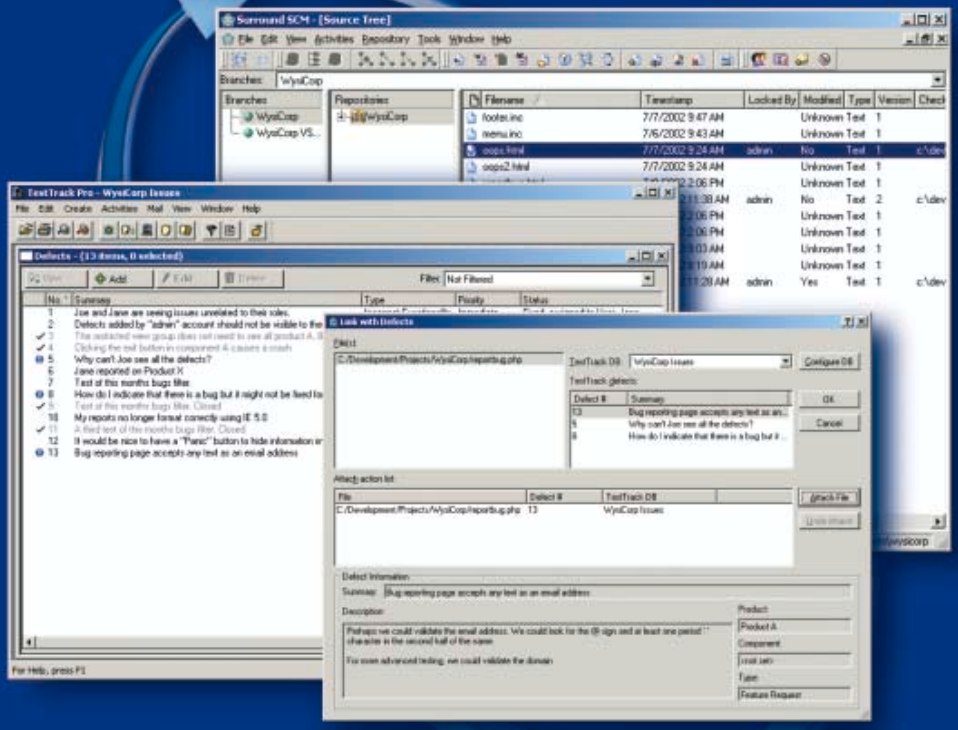
ZERO G

YOUR SOFTWARE DEPLOYMENT PARTNER

WWW.ZEROG.COM

Manage Change
Manage Defects
Effortlessly

Seapine Software™
changing the world of software development



The ultimate change management solution for busy software development teams

Benefits:

- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Fast remote access to your source files and defects — work from anywhere
- Advanced branching makes it a breeze to manage multiple versions of your products

Features:

- Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget
- Seamlessly integrate defect tracking and change management within the Visual Studio IDE

Effective source code control and bug tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

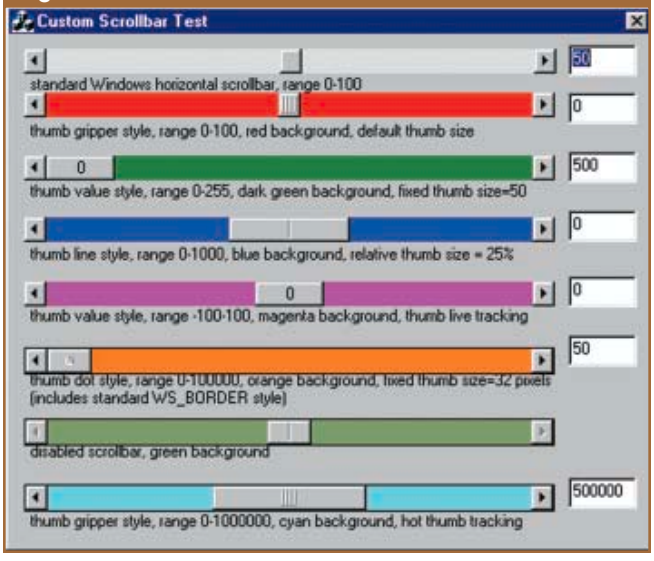
Seapine **Surround SCM**

Seapine **TestTrack PRO**

Download Surround SCM and TestTrack Pro at www.seapine.com or call 1-888-683-6456

Table of Contents

Figure 2 Custom scrollbar test



Listing 2 ScrollBarExt.h: header file

```
//////////////////////////////////////////////////////////////////////////////
//
//    ScrollBarEx.h : header file
//
//    Copyright (c) 2002-2003 Barefoot Productions, Inc.
//    All Rights Reserved.
//
//    Author: Don Metzler
//
//////////////////////////////////////////////////////////////////////////////

#ifndef _SCROLLBAREX_H_
#define _SCROLLBAREX_H_

#include <uxtheme.h>
#include <tmschema.h>

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

const    STYLE_VERTICAL             = 0 ;
const    STYLE_HORIZONTAL        = 1 ;

const    BUTTON_LEFT                 = 0 ;
const    BUTTON_TOP                  = 0 ;
const    BUTTON_RIGHT             = 1 ;
const    BUTTON_BOTTOM             = 1 ;

const    SCROLLAREARECT_ALL         = 0 ;
const    SCROLLAREARECT_LEFT         = 1 ;
const    SCROLLAREARECT_TOP         = 1 ;
const    SCROLLAREARECT_RIGHT    = 2 ;
const    SCROLLAREARECT_BOTTOM    = 2 ;

const    TIMER_MOUSECHECK         = 3 ;
const    TIMER_SCROLLBAR             = 5 ;
const    TIMER_FLASHFOCUS         = 6 ;

const    WM_SCROLLBARPOSCHANGE    = (WM_USER + 1200) ;

//////////////////////////////////////////////////////////////////////////////
// CScrollBarEx window

class CScrollBarEx : public CWnd
{
// Construction
public:

    CScrollBarEx() ;

    virtual BOOL PreTranslateMessage(MSG* inMsg) ;

    //============================================================
    //    Enumerations
    //============================================================

    public:
```

Listing 2 Continued

```
        enum EScrollBarArea
        {
            SCROLLAREA_UNKNOWN = -1,             // this should never occur

            // horizontal scrollbar
            SCROLLAREA_LEFTBUTTON = 0,             // left scroll button
            SCROLLAREA_RIGHTBUTTON = 1,             // right scroll button
            SCROLLAREA_SCROLLAREALEFT = 2,         // left of thumbslider
            SCROLLAREA_SCROLLAREARIGHT = 3,         // right of thumbslider

            SCROLLAREA_THUMBSLIDER = 10,         // thumb slider

            SCROLLAREA_TOTAL = 20,                 // the entire scroll area (between the scroll buttons)

            // vertical scrollbar
            SCROLLAREA_TOPBUTTON = 30,             // top scroll button
            SCROLLAREA_BOTTOMBUTTON = 31,         // bottom scroll button
            SCROLLAREA_SCROLLAREATOP = 32,         // top of thumbslider
            SCROLLAREA_SCROLLAREABOTTOM = 33,     // bottom of thumbslider
        };

        enum EScrollThumbStyle
        {
            SCROLLTHUMBSTYLE_PLAIN = 0,
            SCROLLTHUMBSTYLE_GRIPPER = 1,
            SCROLLTHUMBSTYLE_DOT = 2,
            SCROLLTHUMBSTYLE_LINE = 3,
            SCROLLTHUMBSTYLE_VALUE = 4
        };

        enum EThumbSliderSize
        {
            THUMBSLIDERSIZE_DEFAULT = 0,
            THUMBSLIDERSIZE_FIXEDPIXELS = 1,
            THUMBSLIDERSIZE_RELATIVEPCT = 2
        };

// Attributes
public:

    BOOL             EnableScrollBar(UINT nArrowFlags = ESB_ENABLE_BOTH) ;
    void             EnableWindow(BOOL bEnable = TRUE) ;

    long             GetDefaultThumbSliderWidth() ;
    long             GetScrollPos() ;
    void             GetScrollRange(long& outMinPos, long& outMaxPos) ;
    long             GetThumbSliderWidth() ;

    void             SetBackgroundColor(COLORREF inBackgroundColor) ;
    void             SetHotTrackThumb(BOOL bHotTrackThumb) ;
    void             SetLineSize(long inLineSize) ;
    void             SetPageSize(long inPageSize) ;
    void             SetScrollPos(long inScrollPos, BOOL inRedraw = TRUE) ;
    void             SetScrollRange(long inMinPos, long inMaxPos, BOOL inRedraw = TRUE) ;
    void             SetThumbSliderColor(COLORREF inThumbSliderColor) ;
    void             SetThumbSliderTrackColor(COLORREF inThumbSliderTrackColor, COLORREF
inThumbSliderTextColor, BOOL inTrackThumbSlider) ;
    void             SetThumbSliderSize(EThumbSliderSize inThumbSliderSize, short inRelativeSize = 50) ;
    void             SetThumbStyle(EScrollThumbStyle inThumbStyle) ;

protected:

    EThumbSliderSize m_ThumbSliderSize ;
    BOOL             m_bHotTrackThumb ;
    short             m_ThumbSliderSizeValue ;
    COLORREF         m_BackgroundColor ;
    COLORREF         m_ThumbSliderColor ;
    COLORREF         m_ThumbSliderTrackColor ;
    COLORREF         m_ThumbSliderTextColor ;
    BOOL             m_bTrackThumbSliderFlag ;
    BOOL             m_bTrackScrollAreaFlag ;
    short             m_Style ;
    long             m_PageSize, m_LineSize ;
    CFont             m_Font[2] ;

    long             m_ScrollMinPos, m_ScrollMaxPos ;
    long             m_ScrollPos ;
    BOOL             m_bScrollButtonEnable[2] ;
    BOOL             m_bScrollButtonDownFlag[2];
    BOOL             m_bThumbSliderDownFlag;
    BOOL             m_bLeftButtonDownFlag;
    BOOL             m_bMouseOver ;
    BOOL             m_bMouseOverScrollBarFlag ;
    EScrollBarArea    m_ScrollBarClickArea ;
    short             m_ScrollThumbClickOffset ;
    BOOL             m_bFastScrollFlag ;
    EScrollThumbStyle m_ThumbStyle ;
    CToolTipCtrl     m_tooltip ;
    CRect             m_ThumbSliderRectOrig ;
    CRect             m_ThumbSliderRectActive ;
```

Listing 2 Continued

```
    BOOL            m_bFlashFocus ;
    BOOL            m_bFlashState ;

    float            GetCurrentPosPercent() ;
    void            GetScrollAreaRect(short inScrollArea, CRect& outScrollAreaRect) ;
    void            GetScrollButtonRect(short inScrollButtonIndex, CRect& outScrollButtonRect) ;
    void            GetThumbSliderRect(CRect& outThumbSliderRect, BOOL inDrawRectArea = FALSE) ;

// Operations
public:

    BOOL            Create(DWORD inStyle, const RECT& inRect, CWnd* inParentWnd, UINT inID) ;

protected:

    void            ChangeScrollPos(long inDeltaPos) ;
    CBitmap*        CreatePatternBitmap(CDC* inDC, COLORREF inBackgroundColor) ;

    void            Draw() ;
    void            DrawThumbDot(CDC* inDC, CRect inThumbSliderRect) ;
    void            DrawThumbGripperMark(CDC* inDC, CRect inThumbSliderRect) ;
    void            DrawThumbLine(CDC* inDC, CRect inThumbSliderRect) ;
    void            DrawThumbValue(CDC* inDC, CRect inThumbSliderRect, COLORREF inThumbSliderTextColor) ;

    virtual void    DrawBackground(CDC* inDC, CRect inDrawRect) ;
    virtual    void    DrawButtons(CDC* inDC, CRect inDrawRect) ;
    virtual void    DrawScrollAreas(CDC* inDC, CRect inDrawRect) ;
    virtual void    DrawThumbSlider(CDC* inDC, CRect inDrawRect) ;

    void            EnableScrollTimer(BOOL inEnableTimerFlag, BOOL inFastTimerFlag = FALSE) ;

    EScrollBarArea    GetMouseArea(CPoint inMousePt) ;
    COLORREF        GetHighlightColor() ;
    CFont*            GetScrollThumbFont(BOOL inThumbSliderScrollingFlag) ;
    COLORREF        GetShadowColor() ;

    void            HandleDragThumbSlider(CPoint inPoint) ;
    void            HandleFlashFocus() ;
    void            HandleScrollRepeat(BOOL inAlterScrollPos) ;

    BOOL            IsMouseOverScrollBar(CPoint inMousePoint) ;
    BOOL            IsMouseOverScrollButton(short inScrollButtonIndex, CPoint inMousePoint) ;
    BOOL            IsMouseOverThumbSlider() ;
    BOOL            IsMouseOverThumbSlider(CPoint inMousePoint) ;
    BOOL            IsMouseScrolling() ;

    void            NotifyParentScrollPosChange() ;

    void            TrackMouse(BOOL bTrack) ;

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CScrollBarEx)
    //}}AFX_VIRTUAL

// Implementation
public:

    virtual ~CScrollBarEx() ;

    // Generated message map functions
protected:
    //{{AFX_MSG(CScrollBarEx)
    afx_msg void OnPaint() ;
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point) ;
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point) ;
    afx_msg void OnMouseMove(UINT nFlags, CPoint point) ;
    afx_msg void OnTimer(UINT nIDEvent) ;
    afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message) ;
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct) ;
    afx_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);
    afx_msg void OnSetFocus(CWnd* pOldWnd);
    afx_msg void OnKillFocus(CWnd* pNewWnd);
    afx_msg UINT OnGetDlgCode();
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnCaptureChanged(CWnd *pWnd);
    //}}AFX_MSG
    afx_msg LRESULT OnMouseLeave(WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // _SCROLLBAREX_H_
```

Listing 1 SVGDemo.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"     "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="600" height="600">

    <defs>
        <path id="textPath" d="M120 90 C130 70 210 50 225 90"/>
        <rect id="rect" width="15" height="15"/>
        <g id="lollipop">
            <circle cx="25" cy="25" r="20" fill="red"/>
            <line x1="25" y1="25" x2="25" y2="105" stroke="red" stroke-width="3"/>
        </g>
    </defs>

    <circle cx="45" cy="25" r="20" fill="red"/>

    <rect x="25" y="55" width="40" height="40" fill="green"/>

    <rect x="25" y="105" rx="5" ry="5" width="40" height="40" fill="blue" stroke="red" stroke-width="3"/>

    <ellipse cx="45" cy="160" rx="20" ry="10" fill="red" fill-opacity=".6"/>

    <line x1="35" y1="180" x2="55" y2="215" stroke="black"/>

    <polyline points="35,225 55, 255 25, 235 55, 265" stroke="blue" fill="none"/>

    <polygon points="35,275 55, 305 25, 285 55, 315" stroke="blue" fill="none"/>

    <path d="M15,325 C25,345 45,345 65,305" stroke="red" fill="none"/>

    <use x="15" y="350" fill="blue" xlink:href="#rect"/>
    <use x="45" y="370" fill="green" xlink:href="#rect"/>

    <use x="15"  y="400"  xlink:href="#lollipop"/>

    <text x="120" y="10" fill="red" text-anchor="left">Left</text>

    <text x="120" y="30" fill="red" text-anchor="middle">Middle</text>

    <text x="120" y="50" fill="red" text-anchor="end">End</text>

    <use xlink:href="#textPath" stroke="blue" fill="none"/>
    <text fill="red">
        <textPath xlink:href="#textPath">Following a set path</textPath>
    </text>

    <rect x="200" y="200" width="40" height="40" fill="blue">
        <animate attributeName="width" attributeType="XML" begin="0s" dur="5s" from="40" to="120"
fill="freeze"/>
        <animateMotion path="M200, 300 C 250, 200 300, 400 350, 300 Z" dur="8s" repeatCount="indefinite" />
    </rect>

    <circle cx="120" cy="200" r="25" fill="red">
        <animateColor attributeName="fill" begin="0s" dur="2s" from="red" to="green"
          repeatCount="indefinite"/>
        <animateMotion from="0, 0" to="100, 100" begin="0s" dur="1s" repeatCount="indefinite" />
    </circle>
</svg>
```

Figure 2 The Client interface

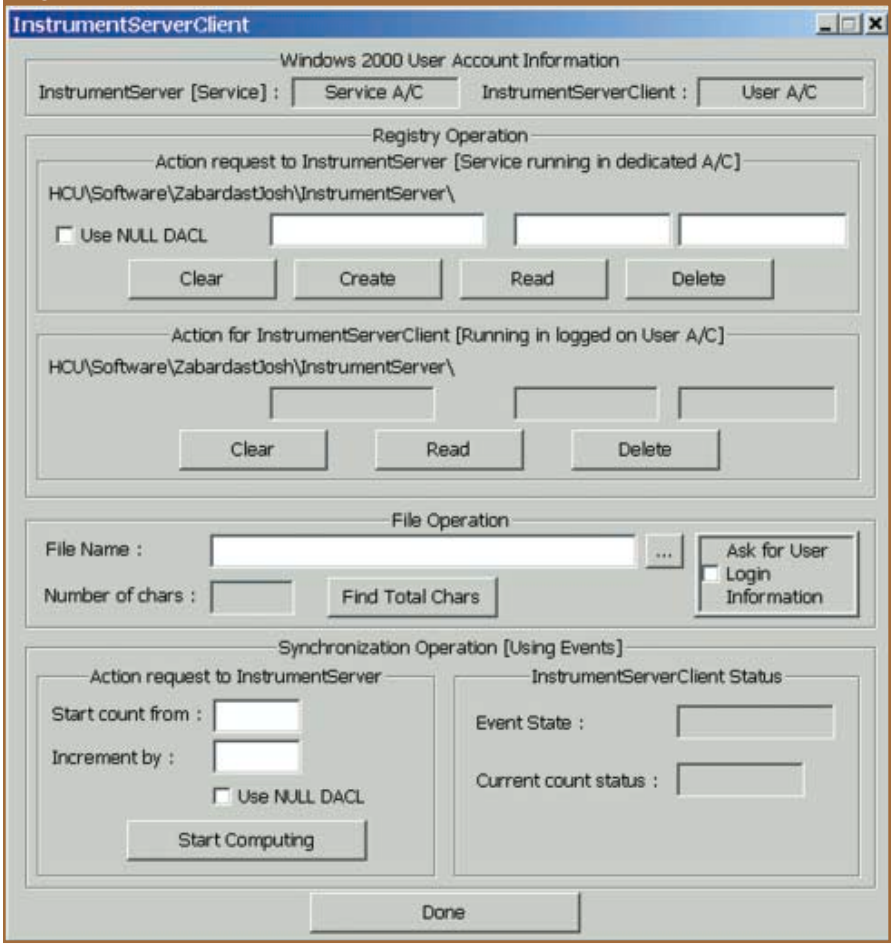


Figure 3 Status of both client and server accounts

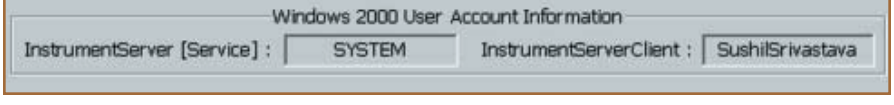


Figure 4 The InformationHandler object

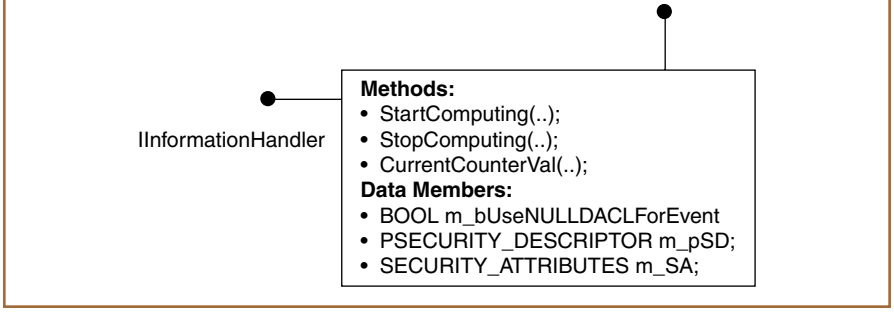


Figure 5 Event sequence for client and server communication

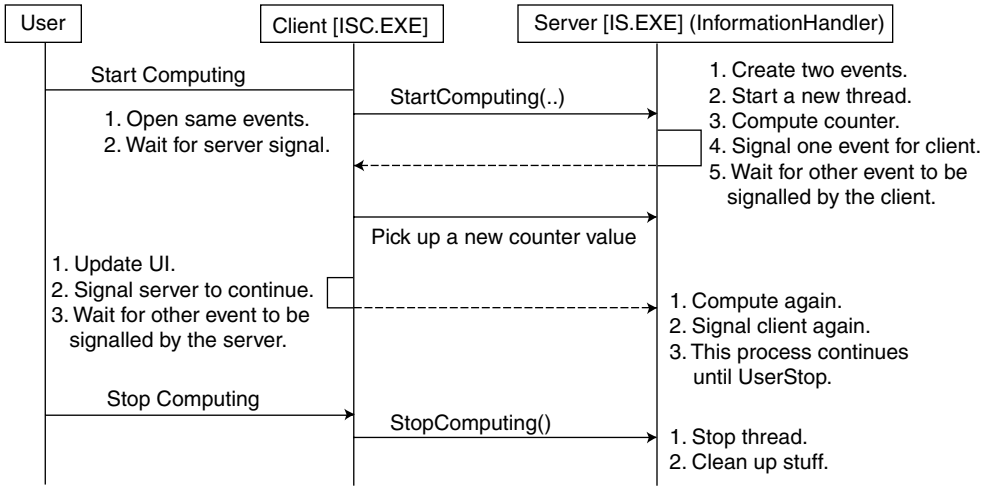


Figure 6 The RegistryHandler object

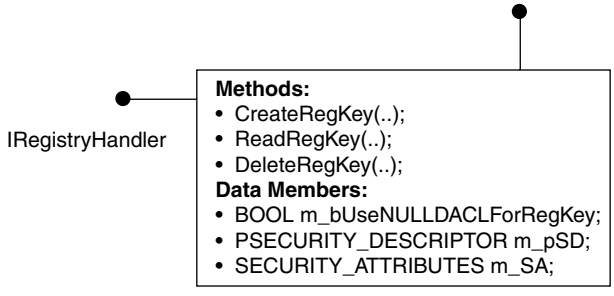


Figure 8 The FileHandler object

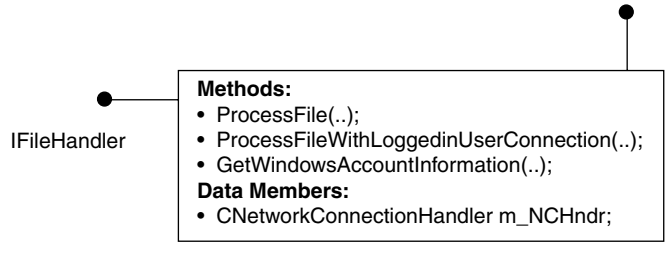


Figure 9 Processing a "Readme.txt" file

Listing 1 A Win32 CreateEvent(..) method definition

```
HANDLE CreateEvent( LPSECURITY_ATTRIBUTES lpEventAttributes, BOOL bManualReset, BOOL bInitialState, LPCTSTR lpName  )
```

Listing 2 Creating a NULL DACL

```
PSECURITY_DESCRIPTOR m_pSD; //Security Descriptor
SECURITY_ATTRIBUTES m_SA;  // Security Attribute

m_pSD = (PSECURITY_DESCRIPTOR) new SECURITY_DESCRIPTOR; // alloc momery for SD
InitializeSecurityDescriptor(m_pSD, SECURITY_DESCRIPTOR_REVISION); //init SD
SetSecurityDescriptorDacl(m_pSD, TRUE, (PACL) NULL, FALSE); // Set NULL DACL
m_SA.nLength = sizeof(m_SA);
m_SA.lpSecurityDescriptor = m_pSD;
m_SA.bInheritHandle = TRUE;

Hnd1 = CreateEvent(&m_SA, TRUE, FALSE, _T("InstrumentServerComputationDoneEvent"));
Hnd2 = CreateEvent(&m_SA, TRUE, FALSE,  _T("InstrumentServerComputationStartEvent"));
```

Listing 3 Events are created with default security

```
Hnd1 = CreateEvent(NULL, TRUE, FALSE, _T(ìInstrumentServerComputionDoneEventíî));
Hnd2 = CreateEvent(NULL, TRUE, FALSE, _T(ìInstrumentServerComputationStartEventîî));
```

Listing 4 Calling the CRegKey class Create method

```
 if (m_pSD)
   eRet = regkey.Create(HKEY_CURRENT_USER, strMainRegkey, REG_NONE, REG_OPTION_NON_VOLATILE,
      KEY_ALL_ACCESS, &m_SA, &dw);
else
   eRet = regkey.Create(HKEY_CURRENT_USER, strMainRegkey, REG_NONE, REG_OPTION_NON_VOLATILE,
      KEY_ALL_ACCESS, NULL, &dw);
```

Listing 5 Snippet from the FileHandler object

```
BOOL CFileHandler::IsUNC(CString strShareName)
{
    DWORD dwBufferSize = 1024;
    TCHAR szBuffer[1024];
    UNIVERSAL_NAME_INFO *puni = (UNIVERSAL_NAME_INFO *) &szBuffer;
    DWORD retval;
    BOOL bUNC=FALSE;

    retval = WNetGetUniversalName((LPCTSTR) strShareName, UNIVERSAL_NAME_INFO_LEVEL, (void *) &szBuffer,
      &dwBufferSize);
    if (retval == NO_ERROR)// if network drive is mapped
    {
        // this is a network file, use the UNC string
        strShareName = puni->lpUniversalName;
        bUNC = TRUE;
    }
    else // if network drive is not mapped but user has a access to some share
        if (retval == ERROR_BAD_DEVICE)
        {
            // this is a network file
            bUNC = TRUE;
        }
        else // if a local drive
            if (retval == ERROR_NOT_CONNECTED)
            {
                bUNC = FALSE;
            }
            else
            {
                // lets see what error we got
                CString msg;
                msg.Format(
                    _T("CFileHandler::IsUNC() WNetGetUniversalName() returned error %d, GetLastError() =
                       %d\n"),
                    retval, GetLastError());
                ATLTRACE(msg);
                bUNC = FALSE;
            }

    return bUNC;
}
```

Listing 6 Snippet from the CNetworkConnectionHandler class

```
SCODE CNetworkConnectionHandler::MapNetworkDrive(CString &strShare, CString &strUserName, CString
  &strPassword)
{
    NETRESOURCE nr;
    TCHAR strShareName[256], strAccessName[256];
    DWORD BufferSize=255, Result;
    int val;


    val = strShare.ReverseFind('\\');
    wcsncpy(strShareName , strShare, val);
    strShareName[val] = NULL;

    CString strt1(strShareName);
    if (CheckIsThisShareAlreadyExist(strt1) == S_OK)
        return S_OK;

    nr.dwType = RESOURCETYPE_DISK;
    nr.lpLocalName = NULL;
    nr.lpRemoteName =  strShareName;
    nr.lpProvider = _T("");


    SCODE retval, retvaldefault;
    retval = WNetUseConnection(NULL, &nr, strPassword, strUserName, CONNECT_REDIRECT, strAccessName,
      &BufferSize, &Result);
    if (retval == NO_ERROR)
    {
        CString strA(strAccessName), strS(strShareName);
        SetAssignedLocalDrive(strA);
        AddNewNetworkConnectionToTheNetworkDriveList(strS);
    }
    else
    {
    ATLTRACE(_T("CNetworkConnectionHandler::WNetUseConnection failed with suppied User Name & Password return
      SCODE = %x\n"), retval);

    }

    return retval;

}
```