THE MICROSOFT JOURNAL FOR DEVELOPERS

FEBRUARY 2011 VOL 26 NO 2

msdn magazine

DYNAMIC PROGRAMMING

COLUMNS

Microsoft

msdn® magazine

FEBRUARY 2011 VOLUME 26 NUMBER 2

LUCINDA ROWLEY Director
DIEGO DAGUM Editorial Director/mmeditor@microsoft.com
KERI GRASSL Site Manager

KEITH WARD Editor in Chief/mmeditor@microsoft.com
TERRENCE DORSEY Technical Editor
DAVID RAMEL Features Editor
WENDY GONCHAR Managing Editor
KATRINA CARRASCO Associate Managing Editor

SCOTT SHULTZ Creative Director
JOSHUA GOULD Art Director

CONTRIBUTING EDITORS K. Scott Allen, Dino Esposito, Julie Lerman, Juval Lowy, Dr. James McCaffrey, Ted Neward, Charles Petzold, David S. Platt

RedmondMediaGroup

Henry Allain President, Redmond Media Group
Matt Morollo Vice President, Publishing
Doug Barney Vice President, Editorial Director
Michele Imgrund Director, Marketing
Tracy Cook Online Marketing Director

ADVERTISING SALES: 508-532-1418/mmorollo@1105media.com

Matt Morollo VP, Publishing
Chris Kourtoglou Regional Sales Manager
William Smith National Accounts Director
Danna Vedder Microsoft Account Manager
Jenny Hernandez-Asandas Director Print Production
Serena Barnes Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA INC

Neal Vitale President & Chief Executive Officer
Richard Vitale Senior Vice President & Chief Financial Officer
Michael J. Valenti Executive Vice President

Abraham M. Langer Senior Vice President, Audience Development & Digital Media
Christopher M. Coates Vice President, Finance & Administration
Erik A. Lindgren Vice President, Information Technology & Application Development
Carmel McDonagh Vice President, Attendee Marketing
David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

MSDN Magazine (ISSN 1528-4859) is published monthly by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. $35.00, International $60.00. Annual digital subscription rates payable in U.S. funds are: U.S. $25.00, International $25.00. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: *MSDN Magazine*, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or IMS/NJ. Attn: Returns, 310 Paterson Plank Road, Carlstadt, NJ 07072.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o *MSDN Magazine*, 16261 Laguna Canyon Road, Ste. 130, Irvine, CA 92618.

Legal Disclaimer: The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@1105media.com

Reprints: For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International, Phone: 212-221-9595, E-mail: 1105reprints@parsintl.com, www.magreprints.com/QuickQuote.asp

List Rental: This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Merit Direct. Phone: 914-368-1000; E-mail: 1105media@meritdirect.com; Web: www.meritdirect.com/1105

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.

Microsoft®

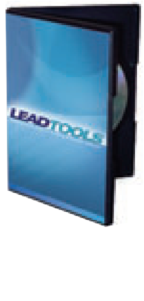
BPA WORLDWIDE

Printed in the USA

Your best source for software development tools!

programmer's paradise®

LEADTOOLS Document Imaging SDK v17.0



by LEAD Technologies

LEADTOOLS Document Imaging has every component you need to develop powerful image-enabled business applications including specialized bi-tonal image processing, document clean up, annotations, high-speed scanning, advanced compression (CCITT G3/G4, JBIG2, MRC, ABC), and Win32/64 binaries for C/C++, .NET, Silverlight, WPF, WCF, & WF. Available add-ons include:

- Multi-threaded OCR/ICR/OMR/MICR/Barcodes (1D/2D)
- Forms Recognition/Processing
- Print Capture and Document Writers
- PDF, PDF/A and XPS

Paradise #
L05 03301A01
$2,007.99

programmers.com/LEAD

IDM UltraEdit

The #1 Best Selling Text Editor in the World

by IDM

UltraEdit is the world's standard in text editors. Millions use UltraEdit as the ideal text/hex/programmers editor on any platform — Windows, Linux, or Mac!

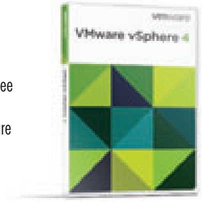
Features include syntax highlighting for nearly any programming language; powerful Find, Replace, Find in Files, and Replace in Files; FTP support, sort, column mode, hex, macros/scripting, large file handling (4+ GB), projects, templates, Unicode, and more.

Named User
1-24 Users
Paradise #
I84 01201A01
$59.95

programmers.com/idm

VMware vSphere Essentials Kit Bundle

vSphere Essentials provides an all-in-one solution for small offices to virtualize three physical servers for consolidating and managing applications to reduce hardware and operating costs with a low up-front investment. vSphere Essentials includes:

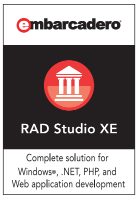


- VMware ESXi and VMware ESX (deployment-time choice)
- VMware vStorage VMFS
- Four-way virtual SMP
- VMware vCenter Server Agent
- VMware vStorage APIs/VCB
- VMware vCenter Update Manager
- VMware vCenter Server for Essentials

for 3 hosts
Paradise #
V55 85101C02
$446.99

programmers.com/vSphere

Embarcadero RAD Studio XE


embarcadero
RAD Studio XE
Complete solution for Windows®, .NET, PHP, and Web application development

NEW RELEASE!

by Embarcadero

Embarcadero RAD Studio XE is a comprehensive application development suite and the fastest way to visually build GUI-intensive, data-driven applications for Windows, .NET, PHP and the Web. RAD Studio includes Delphi®, C++Builder®, Delphi Prism™, and RadPHP™; providing powerful compiled, managed and dynamic language support, heterogeneous database connectivity, rich visual component frameworks, and a vast 3rd party ecosystem – enabling you to deliver applications up to 5x faster across multiple Windows, Web, and database platforms.

Paradise #
CGI 15401A01
$1,383.99

programmers.com/embarcadero

Enterprise Architect Corporate Edition

Visualize, Document and Control Your Software Project

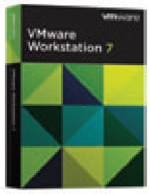
by Sparx Systems

Enterprise Architect is a comprehensive, integrated UML 2.1 modeling suite providing key benefits at each stage of system development. Enterprise Architect 7.5 supports UML, SysML, BPMN and other open standards to analyze, design, test and construct reliable, well understood systems. Additional plug-ins are also available for Zachman Framework, MODAF, DoDAF and TOGAF, and to integrate with Eclipse and Visual Studio 2005/2008.

1-4 Licenses
Paradise #
SP6 03101A02
$182.99

programmers.com/sparxsystems

VMware Workstation 7



VMware Workstation 7 is the gold-standard virtualization software for desktop and laptop computers, with the richest feature set and broadest platform support available. VMware Workstation enables users to create and host multiple virtual machines on a single desktop, expanding the power of desktop systems for IT administrators; software development and test engineers; technical sales, training and support staff; and PC enthusiasts.

VMware Workstation transforms the way technical professionals develop, test, demo, and deploy software. Workstation's innovative features for VMware environments help to reduce hardware cost, save time, minimize risk, and streamline tasks that save time and improve productivity.

for Linux & Windows
Paradise #
V55 22301A04
$149.99

programmers.com/vmware

ActiveReports 6

NEW VERSION 6!

by GrapeCity

The de facto standard reporting tool for Microsoft Visual Studio.NET

- Fast and Flexible reporting engine
- Flexible event-driven API to completely control the rendering of reports
- Wide range of Export and Preview formats including Windows Forms Viewer, Web Viewer, Adobe Flash and PDF
- XCopy deployment
- Royalty-Free Licensing for Web and Windows applications

Professional Ed.
Paradise #
D03 04301A01
$1,310.99

programmers.com/grapecity

Microsoft Visual Studio Professional 2010 Upgrade

by Microsoft

Upgrade to Microsoft Visual Studio 2010 Professional — the integrated environment that simplifies creating, debugging and deploying applications. Unleash your creativity and bring your vision to life with powerful design surfaces and innovative collaboration methods for developers and designers. Work within a personalized environment, targeting a growing number of platforms, including Microsoft SharePoint and cloud applications and accelerate the coding process by using your existing skills. Integrated support for Test-First Development and new debugging tools let you find and fix bugs quickly and easily to ensure high quality solutions.

with MSDN
Single user
Paradise #
M47 40201B02
$488.99

programmers.com/microsoft

TX Text Control 16.0

Word Processing Components

TX Text Control is royalty-free, robust and powerful word processing software in reusable component form.

New Version Released!

- .NET WinForms and WPF rich text box for VB.NET and C#
- ActiveX for VB6, Delphi, VBScript/HTML, ASP
- File formats DOCX, DOC, RTF, HTML, XML, TXT
- PDF and PDF/A export, PDF text import
- Tables, headers & footers, text frames, bullets, structured numbered lists, multiple undo/redo, sections, merge fields, columns
- Ready-to-use toolbars and dialog boxes

Professional Edition
Paradise #
T79 12101A01
$1,109.99

Download a demo today.

programmers.com/textcontrol

New Intel Visual Fortran Compiler

NEW RELEASE!

by Intel

Intel® Visual Fortran Composer XE 2011 includes the latest generation of Intel® Fortran compilers, Intel® Visual Fortran Compiler XE 12.0 for Windows. Intel® Fortran Composer XE is available for Linux and Mac OS X. This package delivers advanced capabilities for development of application parallelism and winning performance for the full range of Intel® processor-based platforms and other compatible platforms. It includes the compiler's breadth of advanced optimization, multithreading, and processor support, as well as automatic processor dispatch, vectorization, and loop unrolling.

for Windows
Single (SSR)
Paradise #
I23 86101E03
$263.99

programmers.com/intel

Microsoft SQL Server Developer Edition 2008 R2

by Microsoft

SQL Server 2008 Developer enables developers to build and test applications that run on SQL Server on 32-bit, ia64, and x64 platforms. SQL Server 2008 Developer includes all of the functionality of Enterprise Edition, but is licensed only for development, test, and demo use. The license for SQL Server 2008 entitles one developer to use the software on as many systems as necessary. For rapid deployment into production, instances of SQL Server 2008 Developer can easily be upgraded to SQL Server 2008 Enterprise without reinstallation.

2-bit/x64
IA64 DVD
Paradise #
M47 31101A04
$41.99

programmers.com/microsoft

Win an iPad!

Place an Order for Software (or Hardware) with Programmer's Paradise and You'll be Entered for a Drawing to Win an iPad Wi-Fi 32GB.

Just Use the **Offer Code TRWD02** When You Place Your Order Online or with Your Programmer's Paradise Representative.

866-719-1528 **programmersparadise.com**

Prices subject to change. Not responsible for typographical errors.

Editor's Note

KEITH WARD

Quick Guide to Getting Published

If there's one question I get asked more than any other at conferences, it's this: "Hey, I have a great idea for an *MSDN Magazine* article. How do I get it published?"

It's a good question. After all, writing a magazine article isn't like writing code; forgetting the semicolon I just placed (after "code") is less serious than forgetting a semicolon in C#. In one important way, however, they are similar—they're both creative endeavors in which you start with a blank page or empty Visual Studio project window. From that void comes, hopefully, something wonderful. Here are the most important things to do.

1. Bribes never hurt. Remember the rule of thumb: The more zeros in the check, the more likely the acceptance of your article. All proceeds go to support needy children—mine.

OK. Enough with the comedy portion of our show. Please don't send me cash.

The first thing you *do* need, however, is a great idea for an article. You could have Bill Gates' programming savvy and Hemingway's writing chops, but if your idea's as lame as the new crop of reality TV shows, it doesn't matter.

So what's a "great" idea, you properly ask? That's hard to quantify, but I can list some general guidelines. First, as any editor will tell you, *read back issues*. If you're a regular *MSDN Magazine* reader and pore over every issue, you'll likely have an intuitive sense of the kind of articles we publish. If not, get your hands on the last half-dozen issues or so (remember that they're published online in their entirety as well) and do some research.

Such research will quickly reveal some important facts. No. 1, this is a very technical magazine written mostly for experienced developers. Introductory and overview type articles don't make it in, for the most part. These are not 100-level stories, so keep that in the front of your mind. There are plenty of places to publish those types of articles, and they're of great value to the right audience. Just know that it's not *our* audience.

Fact No. 2: Articles of a general nature are shunned. Another rule of thumb: General = Bad. Specific = Good. "I'd like to write an article on Windows Workflow Foundation" will get a thumbs-down from yours truly. "I'd like to write an article on authoring custom control flow activities in WF 4," on the other hand, is much more likely to get a thumbs-up. In fact, it did for Leon Welicki, who wrote that exact article for the last issue.

Keep in mind that our readers are generally high-level pros who want specific answers to specific development problems or challenges. To that end, present practical advice for solving real-world issues. I don't need articles on what the OData Protocol is or can be used for: instead, pitch me an idea on integrating OData with existing Atom- and AtomPub-based readers and writers, as Chris Sells did in the August 2010 issue.

Another source of great anxiety for would-be writers is writing experience. They wonder if they have a chance if they've published nothing but a few blog entries here and there. If that's you, have no fear: previous writing experience is not a prerequisite. I'm more interested in the article idea and how you'd execute it than in how much writing you've done.

Certainly it doesn't hurt if you've written before, but if you can demonstrate basic writing competency in your communications—the initial proposal, as well as additional correspondence—we'll consider hiring you.

If you're an inexperienced writer, know from the outset that writing is *difficult*, and writing these types of articles is an art unto itself. It's one thing to know how to build software, and another thing entirely to write about building software.

And you're not done even then. The editing process that follows the writing—where we work with you to polish, fine-tune or even ask you to make major revisions to your work—can be agonizing.

I don't say this to scare you off, but to give a realistic glimpse of what lies ahead. If you truly want to do this, what I'm saying won't scare you anyway, and you're likely to come out with a successful article and see your name in the pages of *MSDN Magazine*.

Send article ideas to me at mmeditor@microsoft.com.

Keith Ward

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in MSDN Magazine are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN, and Microsoft logos are used by 1105 Media, Inc. under license from owner.

OnTime Now!

Cloud-based Scrum Tool for Developers

In just seconds, get your team started with the world's most advanced agile project management solution. See why *OnTime Now* is so popular with agile dev teams!

$39 / month for 5 users • Free 30-day Team Trial

axosoft

Software for Software Development

Subversion for Windows on IIS

Finally, you can utilize Subversion inside a Windows environment. Quick setup and intuitive management make RocketSVN the premiere Subversion solution for your team.

Server: 2 users Free • $79 each add'l user
Visual Studio Client: $49/user

Local Scrum Tool for Developers

Ship Software on time with the locally-installed award-winning project management tool for agile development teams. *MS-SQL* server back-end.

Starts @ $395 for teams of 5 • Free for 1 user

www.axosoft.com
(800) 653-0024 • (480) 362-1900

TERRENCE DORSEY

Windows Phone 7 Development Tools and Resources

Without a doubt, one of the hottest areas for development these days is mobile apps. The smartphone market is expected to grow by 55 percent in the next year alone.

Last year, Microsoft dove back into the mobile device fray with the Windows Phone 7 platform. For the consumer, Windows Phone 7 offers a modern multi-touch interface, integration with e-mail, Facebook and Zune services, and the ability to install apps to have fun or get work done. For the developer, Windows Phone 7 provides a straightforward programming target based on Silverlight and the Microsoft .NET Framework, along with the ability to reach a large—and growing—audience of customers.

Sounds like a no-brainer, right?

We've already shown you a few useful tips for Windows Phone 7 programming in the pages of *MSDN Magazine*, but here's a concise guide to some of the tools and information you'll need to get started and create useful, polished apps.

At the sound of the tone, it's time to get coding ...

Windows Phone Developer Tools

The basic tools you'll need to start writing code for Windows Phone 7 are Visual Studio 2010 and Windows Phone Developer Tools (**microsoft.com/express/Phone**).

The developer tools include Visual Studio 2010 Express for Windows Phone, the Windows Phone Emulator, Silverlight for Windows Phone, XNA Game Studio 4.0 and Expression Blend 4 for Windows Phone. If you already have Visual Studio 2010, XNA Game Studio or Expression Blend 4 installed, the developer tools will just update the applications with templates and other elements you need for Windows Phone development.

Once you have the tools installed, start browsing through the MSDN Windows Phone Development documentation (**bit.ly/fSYt8d**). This is a crucial resource that you'll come back to again and again. Add it to your favorites today.

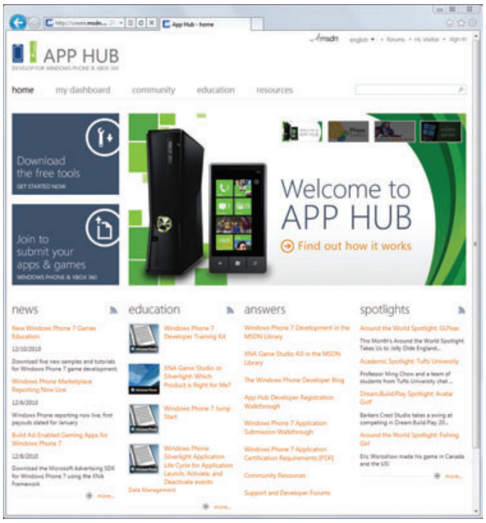


Figure 1 **App Hub**

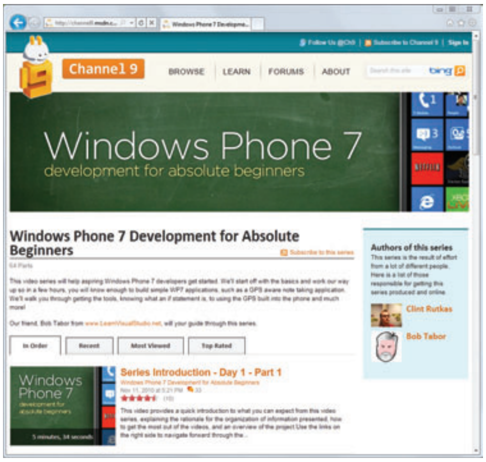


Figure 2 **Windows Phone 7 for Absolute Beginners**

App Hub

Next, head over to the Microsoft App Hub (**create.msdn.com**) and set up your account. App Hub is a community for developers of apps and games for Windows Phone 7 or games for Xbox 360. You can find tools, advice and the support of fellow app developers here. Plus, you'll use the App Hub dashboard as the official tool for submitting your apps for the Windows Phone marketplace.

To register for App Hub you'll need a Windows Live ID, and there's a yearly $99 fee for membership.

Visual Basic Development

The first tools for Windows Phone development focused on C# as the coding language. But if you prefer Visual Basic, don't fret. The Visual Studio team has released Visual Basic for Windows Phone Developer Tools (**bit.ly/haiyqH**), which allows you to dive right in.

The team announcement even includes a tutorial for creating your first Visual Basic app for Windows Phone 7, walking you through the development of a simple tip-calculator app. No excuses now—get coding!

Windows Phone 7 for Absolute Beginners

Still not sure what to do next? Get yourself over to Channel 9 and check out the "Windows Phone 7 Development for Absolute Beginners" video series (**bit.ly/fZJSqC**). Clint Rutkus and Bob Tabor (from **LearnVisualStudio.net**) guide you through this series of 65 short videos covering everything from platform basics to debugging and error handling. You'll even dive into details such as using the GPS capabilities of Windows Phone 7 devices.

XPF: A Layout Framework for XNA

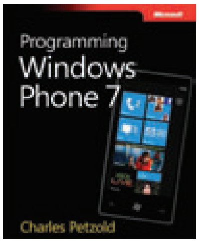
General UI design and composition on Windows Phone 7 utilizes Windows Presentation Foundation (WPF) and Silverlight. But if you're building games for the phone platform, you're most likely going to

be using XNA Game Studio (**msdn.microsoft.com/aa937791**), which uses a different layout architecture.

To simplify design under XNA, Red Badger has created XPF (**bit.ly/dMAIOA**), a layout framework for creating Windows and Windows Phone 7 apps with XNA. The XPF framework runs in a pure XNA application, yet was written to be familiar to WPF and Silverlight developers. There are 11 built-in controls, along with support for dependency properties, attached properties, animation and data binding. Plus, XPF is extensible to accommodate your own custom controls.

It's also worth taking a look through the Red Badger blog for additional tips and tricks for building and testing your Windows Phone 7 applications (**red-badger.com/Blog/?tag=/wp7**).

Free Charles Petzold eBook



Programming Windows Phone 7 eBook

Our own Charles Petzold (**bit.ly/enl2dg**), along with the Windows Phone 7 team, wrote the aptly titled "Programming Windows Phone 7" (**bit.ly/foFbvZ**). As you'd expect from the guy who wrote five editions of the seminal "Programming Windows" (Microsoft Press, 1998), this tome is extensive and definitive. The best part? It's free. Download the book in PDF format along with all of the sample code. For ongoing discussion of the book and Windows Phone 7 development issues, read Petzold's blog as well (**charlespetzold.com/blog/blog.xml**).

And next time someone asks you a Windows Phone programming question, you can still tell them: "Look it up in Petzold."

31 Days of Windows Phone 7

In the same vein as the Channel 9 videos mentioned earlier, Microsoft developer evangelist Jeff Blankenburg blogged "31 Days of Windows Phone 7" (**bit.ly/dJ0u8k**), writing a post every day in October 2010 that covers a different aspect of coding for Windows Phone 7. Blankenburg starts with the basic project template and works through topics including storage, UI controls, debugging, and even submitting your app to the Windows Phone Marketplace and integrating ads into your app.

Blankenburg's blog is full of other Silverlight and Windows Phone 7 development tips, and the 31 Days posts have been translated into Russian and Spanish.

Helpful Client Frameworks

Frameworks can really take a lot of the hassle out of building infrastructure or adding specific functionality to your application. Two of the more popular client frameworks for Windows Phone 7 on CodePlex are Caliburn Micro and nRoute.

Caliburn Micro (**caliburnmicro.codeplex.com**) is a small-yet-powerful implementation of the Caliburn (**caliburn.codeplex.com**) framework for Silverlight, Windows Phone 7 and WPF. These frameworks are designed to help you build application UIs quickly and easily using the Model-View-Controller (MVC), Model-View-Presenter (MVP), Model-View-ViewModel (MVVM) and Application Controller patterns.

Caliburn Micro has an active community providing information, support and examples. Get started by checking out the "Caliburn.Micro Soup to Nuts" series of posts on devlicio.us (**bit.ly/hLT7lL**) to follow the creation of a simple Silverlight application, starting with the basics of getting Caliburn Micro from the repository.

nRoute (**nroute.codeplex.com**) is another client framework for Silverlight, Windows Phone 7 and WPF that lets you build apps using the MVVM pattern. John Thiriet has an extensive series of blog posts (**blog.john-thiriet.com/en/**) showing you how to use nRoute—in both French and English—and the orktane blog has a great tutorial on using nRoute to create a game (**bit.ly/fwF9pL**).

One of the great side effects of the Windows Phone 7 platform has been the outpouring of shared information from members of the development community.

More Tips and Tricks

One of the great side effects of the Windows Phone 7 platform has been the outpouring of shared information from members of the development community.

You'll find everything from getting-started guides to tips for handling specific development tasks, and even ways to run other languages on the phone. Here's what we think are some of the best posts out there:

MobiForge
Introduction to Windows Phone 7 Development Series
bit.ly/gmygUR

IE for Windows Phone Team Weblog
Targeting Mobile-Optimized CSS at Windows Phone 7
bit.ly/h5ZJwY

Ivo Manolov
Registering Your WP7 as a Developer Device
bit.ly/exSPzi

Kevin Marshall
WP7 Development Tips Part 1
bit.ly/fTcljh

Mike Ormond
Windows Phone 7 Screen Capture
bit.ly/hSraKE

MSDN Magazine
IronRuby on Windows Phone
bit.ly/gA5x7J

TERRENCE DORSEY *is the technical editor of* MSDN Magazine. *You can read his blog at terrencedorsey.com or follow him on Twitter: @tpdorsey.*

THANKS to the following technical experts for reviewing this article: *Luke Nyswonger, Joshua Partlow, Brendan Mitchell, Orville McDonald*

Full-Featured **Reporting Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/REPORTING**
or **CALL US (818) 844-3383**

Devexpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

All trademarks and registered trademarks are the property of their respective owners.

CUTTING EDGE

DINO ESPOSITO

Policy Injection in Unity

In the past two articles, I presented aspect-oriented programming (AOP) using Microsoft Unity 2.0. Formalized in the 1990s as a way to further improve and complement object-oriented programming (OOP), AOP was recently revamped, and many Inversion of Control (IoC) libraries support it. Unity is no exception. The main purpose of AOP is to let developers more effectively deal with crosscutting concerns. In essence, AOP addresses the following question: When you design an object model for an application, how do you deal with aspects of the code such as security, caching or logging? These aspects are key to implementation, but don't strictly belong to the objects in the model you're building. Should you spoil your design to incorporate non-business aspects? Or are you better off decorating your business-oriented classes with additional aspects? If you choose the latter, AOP basically provides a syntax to define and attach these aspects.

> An aspect is the implementation of a crosscutting concern.

An aspect is the implementation of a crosscutting concern. In the definition of an aspect, you need to specify a few things. First, you need to provide the code for the concern you implement. In AOP jargon, this is known as the advice. An advice is applied to a specific point of code—whether the body of a method, the getter/setter of a property or perhaps an exception handler. This is known as the join point. Finally, in AOP jargon, you find pointcuts. A pointcut represents a collection of join points. Usually, pointcuts are defined by criteria using method names and wildcards. AOP ultimately acts in the runtime to inject the code of the advice before, after and around the join point. An advice then is associated with a pointcut.

In the previous articles, I explored the interception API of Unity. The API lets you define advices to attach to classes. In Unity jargon, the advice is a behavior object. You typically attach the behavior to a type that's resolved via the IoC mechanism of Unity, even though the interception mechanism doesn't strictly require the IoC functionality. In fact, you can configure interception to apply also to instances created via plain code.

A behavior consists in a class that implements a fixed interface—the IInterceptionBehavior interface. The interface features a method named Invoke. By overriding this method, you actually define the steps you want to be executed before or after the regular method call, or both. You can attach a behavior to a type using fluent code as well as a configuration script. In this way, all you do is define a join point. But what about pointcuts?

As we saw last month, all intercepted methods on the target object will execute according to the logic expressed in the Invoke method of the behavior object. The basic interception API doesn't provide you with the ability to distinguish between methods and doesn't support specific matching rules. To get this, you may resort to the policy injection API.

Policy Injection and PIAB

If you've used versions of the Microsoft Enterprise Library (EntLib) prior to the latest version, 5.0, you may have heard about Policy Injection Application Block (PIAB), and chances are that you also took advantage of it in some of your applications. EntLib 5.0 also features a PIAB module. So what's the difference between the Unity policy injection and EntLib PIAB?

In EntLib 5.0, PIAB exists mostly for compatibility reasons. The content of the PIAB assembly changed in the new version. In particular, all the machinery for interception is now part of Unity and all system-provided call handlers in earlier versions of EntLib were moved to other assemblies, as shown in **Figure 1**.

As you can see in **Figure 1**, each call handler was moved to the assembly of the associated application block. So the exception-handling call handler moved to the exception-handling application block and the validation handler moved to the validation-application block and so on. The only exception to this rule was the performance-counter handler, which moved to the PolicyInjection assembly. Although the assemblies changed, the namespace of the classes remained just the same. It's also worth noting that, due to security concerns, the caching call handler previously included in PIAB was removed from EntLib 5.0 and made available only through the EntLib Contrib CodePlex Web site at bit.ly/gIcP6H. The net effect of these changes is that PIAB is now made of legacy components available just for backward compatibility that still require some code changes

Figure 1 **Refactoring of Call Handlers in the Microsoft Enterprise Library 5.0**

Call Handler	New Assembly in Enterprise Library 5.0
Authorization handler	Microsoft.Practices.EnterpriseLibrary.Security.dll
Caching-handling handler	Removed from PIAB
Exception-handling handler	Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.dll
Logging handler	Microsoft.Practices.EnterpriseLibrary.Logging.dll
Performance-counter handler	Microsoft.Practices.EnterpriseLibrary.PolicyInjection.dll
Validation handler	Microsoft.Practices.EnterpriseLibrary.Validation.dll

Powerhouse **Analytics Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/ANALYTICS**
or **CALL US (818) 844-3383**

DevExpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

All trademarks and registered trademarks are the property of their respective owners.

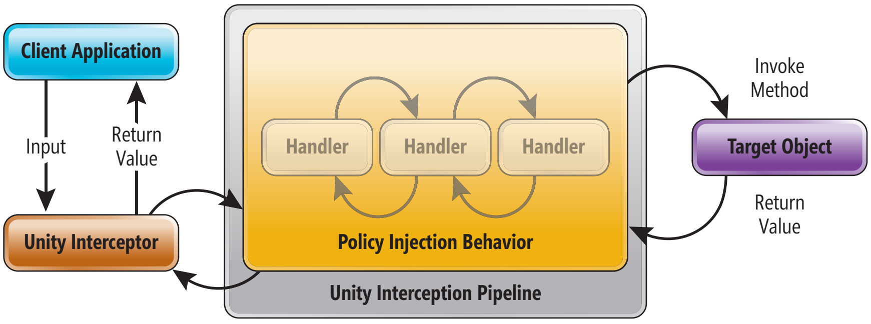


Figure 2 **The Call Handler Pipeline in the Unity Policy Injection**

to compile with version 5.0. Unless you have serious legacy dependencies, the recommended approach is that you upgrade your policy injection layers to take advantage of the new (and largely similar) policy injection API baked into the Unity application block. Let's find out more about policy injection in Unity.

Policy Injection at a Glance

Policy injection is a layer of code that extends the basic Unity interception API to add mapping rules and call handlers on a per-method basis. Implemented as a special interception behavior, policy injection consists of two main phases: initialization and execution time.

During the initialization phase, the framework first determines which of the available policies can be applied to the target method being intercepted. In this context, a policy is described as a set of operations that can be injected in a particular order between the object being intercepted and its actual caller. You can only intercept methods on objects (either existing instances or newly created instances) that have been explicitly configured for policy injection.

Figure 3 **List of Supported Matching Rules in Unity 2.0**

Matching Rule	Description
AssemblyMatchingRule	Selects target objects based on types in the specified assembly.
CustomAttributeMatchingRule	Selects target objects based on a custom attribute at the member level.
MemberNameMatchingRule	Selects target objects based on member name.
MethodSignatureMatchingRule	Selects target objects based on the signature.
NamespaceMatchingRule	Selects target objects based on the namespace.
ParameterTypeMatchingRule	Selects target objects based on the type name of a parameter for a member.
PropertyMatchingRule	Selects target objects based on member names, including wildcard characters.
ReturnTypeMatchingRule	Selects target objects based on the return type.
TagMatchingRule	Selects target objects based on the value assigned to an ad hoc Tag attribute.
TypeMatchingRule	Selects target objects based on the type name.

Having figured out the list of applicable policies, the policy injection framework prepares the pipeline of operations (an operation is referred to as a call handler). The pipeline results from the combination of all the handlers defined for each of the matching policies. The handlers in the pipeline are sorted based on the order of the policy and the priority assigned to each handler in the parent policy. When a policy-enabled method is invoked, the previously built pipeline is processed. If the method, in turn, places calls to other policy-enabled methods on the same object, the handler pipelines of those methods are merged into the main pipeline.

Call Handlers

A call handler is more specific than a "behavior" and really looks like an advice, as it was originally defined in AOP. Whereas a behavior applies to a type and leaves to you the burden of taking different actions for different methods, a call handler is specified on a per-method basis.

Call handlers are composed in a pipeline and invoked in a predetermined order. Every handler is able to access details of the call, including method name, parameters, return values and expected return type. A call handler can also modify parameters and return values, stop the propagation of the call down the pipeline and raise an exception.

It's interesting to note that Unity doesn't come with any call handlers. You can only create your own, or reference application blocks from EntLib 5.0 and use any of the call handlers listed in **Figure 1**.

A call handler is a class that implements the ICallHandler interface, like this:

```
public interface ICallHandler
{
  IMethodReturn Invoke(
    IMethodInvocation input,
    GetNextHandlerDelegate getNext);
  int Order { get; set; }
}
```

The Order property indicates the priority of this handler related to all others. The Invoke method returns an instance of a class that contains any return value from the method.

The implementation of a call handler is quite simple in the sense that it's just expected to do its own specific things and then let the pipeline go. To yield control to the next handler in the pipeline, the handler calls the getNext parameter it receives from the Unity runtime. The getNext parameter is a delegate defined as:

```
public delegate InvokeHandlerDelegate GetNextHandlerDelegate();
```

In turn, the InvokeHandlerDelegate is defined as:

```
public delegate IMethodReturn InvokeHandlerDelegate(
  IMethodInvocation input,
  GetNextHandlerDelegate getNext);
```

The Unity documentation provides a clear diagram that illustrates interception. In **Figure 2**, you see a slightly modified diagram that presents the architecture of policy injection.

Within the boundaries of a system-provided policy injection behavior, you see the chain of handlers to process a given method invoked on a proxy object or derived class. To complete the overview of policy injection in Unity, we need to take a look at matching rules.

Figure 4 **The NonNegativeCallHandler Class**

```
public class NonNegativeCallHandler : ICallHandler
{
  public IMethodReturn Invoke(IMethodInvocation input,
                              GetNextHandlerDelegate getNext)
   {
     // Perform the operation
     var methodReturn = getNext().Invoke(input, getNext);

     // Method failed, go ahead
     if (methodReturn.Exception != null)
       return methodReturn;

     // If the result is negative, then throw an exception
     var result = (Int32) methodReturn.ReturnValue;
     if (result <0)
     {
       var exception = new ArgumentException("...");
       var response = input.CreateExceptionMethodReturn(exception);

       // Return exception instead of original return value
       return response;
     }

     return methodReturn;
   }

    public int Order { get; set; }
}
```

Matching Rules

Through a matching rule, you specify where to apply your interception logic. If you use behaviors, your code applies to the entire object; with one or more matching rules, you can define a filter. A matching rule indicates a criterion to select objects and members to which Unity will attach a handler pipeline. Using AOP terminology, a matching rule is the criterion that you use to define the pointcuts. **Figure 3** lists the matching rules natively supported by Unity.

A matching rule is a class that implements the IMatchingRule interface. Armed with this knowledge, let's see how to work with policy injection. There are essentially three ways in which you can define policies: using attributes, using fluent code and via configuration.

Figure 5 **Fluent Code to Define a Set of Matching Rules**

```
public static UnityContainer Initialize()
{
  // Creating the container
  var container = new UnityContainer();
  container.AddNewExtension<Interception>();

  // Adding type mappings
  container.RegisterType<ICalculator, Calculator>(
    new InterceptionBehavior<PolicyInjectionBehavior>(),
    new Interceptor<TransparentProxyInterceptor>());

  // Policy injection
  container.Configure<Interception>()
    .AddPolicy("non-negative")
    .AddMatchingRule<TypeMatchingRule>(
      new InjectionConstructor(
        new InjectionParameter(typeof(ICalculator))))
    .AddMatchingRule<MemberNameMatchingRule>(
      new InjectionConstructor(
        new InjectionParameter(new[] {"Sub", "Test"})))
    .AddMatchingRule<ReturnTypeMatchingRule>(
      new InjectionConstructor(
        new InjectionParameter(typeof(Int32))))
    .AddCallHandler<NonNegativeCallHandler>(
      new ContainerControlledLifetimeManager(),
        new InjectionConstructor());

  return container;
}
```

Figure 6 **Preparing Policy Injection in the Configuration File**

```
<unity xmlns="http://schemas.microsoft.com/practices/2010/unity">
  <assembly name="PolicyInjectionConfig"/>
  <namespace name="PolicyInjectionConfig.Calc"/>
  <namespace name="PolicyInjectionConfig.Handlers"/>

  <sectionExtension  ... />

  <container>
    <extension type="Interception" />

    <register type="ICalculator" mapTo="Calculator">
      <interceptor type="TransparentProxyInterceptor" />
      <interceptionBehavior type="PolicyInjectionBehavior" />
    </register>

    <interception>
      <policy name="non-negative">
        <matchingRule name="rule1"
          type="TypeMatchingRule">
          <constructor>
             <param name="typeName" value="ICalculator" />
          </constructor>
        </matchingRule>
        <matchingRule name="rule2"
          type="MemberNameMatchingRule">
          <constructor>
            <param name="namesToMatch">
              <array type="string[]">
                <value value="Sub" />
              </array>
            </param>
          </constructor>
        </matchingRule>
        <callHandler name="handler1"
          type="NonNegativeCallHandler">
          <lifetime type="singleton" />
        </callHandler>
      </policy>
    </interception>

  </container>
</unity>
```

Adding Policies via Attributes

Figure 4 shows an example call handler that throws an exception if the result of an operation is negative. I'll be using this same handler in various scenarios.

The simplest way to use the handler is by attaching it to any method where you think it can be useful. For this, you need an attribute, such as:

```
public class NonNegativeCallHandlerAttribute : HandlerAttribute
{
  public override ICallHandler CreateHandler(
    IUnityContainer container)
  {
    return new NonNegativeCallHandler();
  }
}
```

Here's a sample Calculator class that you decorate with attribute-based policies:

```
public class Calculator : ICalculator
{
  public Int32 Sum(Int32 x, Int32 y)
  {
    return x + y;
  }

  [NonNegativeCallHandler]
  public Int32 Sub(Int32 x, Int32 y)
  {
    return x - y;
  }
}
```

The result is that calls to method Sum proceed as usual regardless of the returned value, whereas calls to method Sub will throw an exception if a negative number is returned.

Using Fluent Code

If you don't like attributes, you can express the same logic via a fluent API. In this case, you must provide many more details as far as matching rules are concerned. Let's see how to express the idea that we want to inject code only in methods that return an Int32 and are named Sub. You use the fluent API to configure the Unity container (see **Figure 5**).

Note that if you use the ContainerControlledLifetimeManager manager, you're guaranteed that the same call handler instance is shared by all of the methods.

The effect of the code is that any concrete type that implements ICalculator (that is, is configured to be intercepted and is resolved through Unity) will select two potential candidates for injection: methods Sub and Test. However, only methods with an Int32 return type will survive the further matching rule. This means that, for example, Test will be ruled out if it happens to return a Double value.

Adding Policies Through Configuration

Finally, the same concept can be expressed using the configuration file. **Figure 6** shows the expected content of the <unity> section.

As it turns out, when you have multiple matching rules in a single policy, the final result is the Boolean operator AND applies to all of them (meaning all of them must be true). If you've defined multiple policies, then each one is evaluated for matching—and handlers applied—independently. Therefore, you can get handlers applied from different policies.

Interception at a Glance

To recap, interception is the way in which most IoC frameworks in the Microsoft .NET Framework space implement aspect orientation. Through interception, you're given the chance of running your own code before or after any given method in any given type in any given assembly. EntLib in the past provided a specific application block, PIAB, for achieving this. In EntLib 5.0, the underlying engine of PIAB has been moved into Unity and implemented as a special behavior for the Unity low-level interception API that I covered in my previous two columns. The policy injection behavior requires the use of a Unity container and won't work only through the low-level interception API.

The low-level interception API, however, doesn't let you select the type members you want to intercept; you have to write the code to do that yourself. With the policy injection behavior, though, you can concentrate on the details of the behavior you want, and let the library take care of figuring out which methods it applies to based on the rules you give it. ■

DINO ESPOSITO *is the author of "Programming Microsoft ASP.NET MVC" (Microsoft Press, 2010) and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. You can join his blog at weblogs.asp.net/despos.*

THANKS to the following technical expert for reviewing this article: *Chris Tavares*

DevExpress
DXGrid

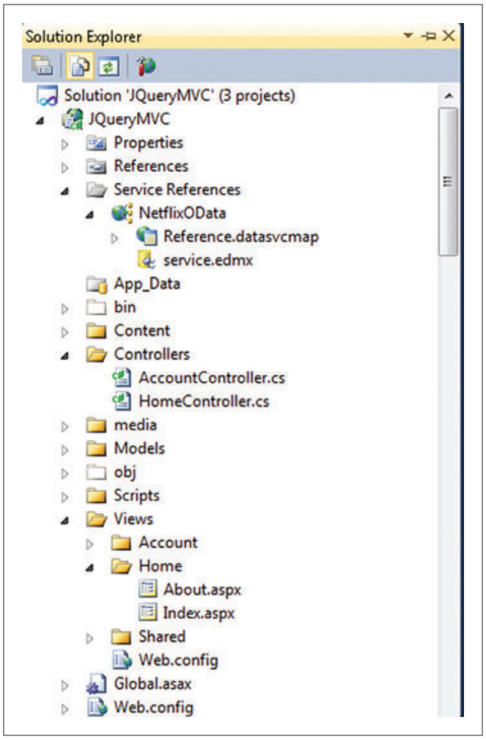
XtraReports
Side-by-Side Reports
Shortcuts
Activities
Calendar
Archived Folders

Slice and Dice OData with the jQuery DataTables Plug-In

The Open Data Protocol (OData) allows data producers to provide their data over the Web in a common format that can be consumed by anyone using an HTTP-enabled technology. Data is provided through URIs and you can use the common HTTP verbs—GET, PUT, POST, MERGE and DELETE—to interact with the data. You can handle this interaction directly through a language like JavaScript or by using a client API such as the Microsoft .NET Framework, Silverlight, PHP or others provided by Microsoft. Either way, you can interact with all OData feeds in the same way.

There are a growing number of publicly provided OData services such as commercial feeds from Netflix Inc. and eBay Inc., World Cup data, even a service that supplies 150 years of baseball statistics.

Accessing data is getting easier and easier, but what about presenting the data? When you have 150 years worth of baseball stats or thousands of movie titles, there's still some effort on the client side to retrieve and navigate through all of that data.



```
Solution Explorer
Solution 'JQueryMVC' (3 projects)
  JQueryMVC
    Properties
    References
    Service References
      NetflixOData
        Reference.datasvcmap
        service.edmx
    App_Data
    bin
    Content
    Controllers
      AccountController.cs
      HomeController.cs
    media
    Models
    obj
    Scripts
    Views
      Account
      Home
        About.aspx
        Index.aspx
      Shared
      Web.config
    Global.asax
    Web.config
```

Figure 1 **The MVC Project in Solution Explorer**

At a recent Vermont .NET User Group presentation on jQuery, I was inspired by a jQuery plug-in called DataTables as a low-investment way to allow users to slice and dice large amounts of data. The power of DataTables is in its blazingly fast client-side processing, although it does allow you to be more interactive with server-side code if you desire.

JQuery is a client-side Web technology (and can be used in any type of Web application) that simplifies working with JavaScript. If you've talked to anyone who's jumped on the jQuery bandwagon, you'll find a lot of passion for the technology. DataTables is one of a huge number of jQuery plug-ins. And you can use jQuery in any type of Web application.

As I happen to do most of my work with the .NET Framework, in this column I'll demonstrate using some of the basic DataTables plug-in features in applications using both ASP.NET MVC and WebForms. However, the logic in the WebForms app will be driven by client-side code. I'll be working with the Netflix OData service (http://odata.netflix.com/v1/Catalog), which gives me an opportunity to show you how to deal with some common pitfalls you may encounter when using various OData services.

You can download the DataTables plug-in from datatables.net. If you're new to consuming OData, you might want to get up to speed by visiting the WCF Data Services section of the MSDN Developer Center at msdn.microsoft.com/data/odata.

Query OData with LINQ and the Client APIs

I'll begin with a simple MVC application where I've added a service reference to http://odata.netflix.com/v1/Catalog using the Visual Studio Add Service Reference wizard. This, in turn, creates proxy classes for me to consume in my application and builds an Entity Data Model based on the service, as shown in **Figure 1**. The wizard also adds references to the .NET Framework OData client library APIs. Both the .NET Framework and Silverlight OData client libraries make working with OData fairly simple thanks to their support of LINQ querying.

My startup controller, HomeController.cs, uses the OData client library and the service proxy to query for all of the movie titles in a particular genre: Independent. The results of the query are returned to the View associated with this particular controller action:

```
public ActionResult Index() {
  var svcUri = new Uri("http://odata.netflix.com//v1//Catalog");

  var context = new NetflixOData.NetflixCatalog(svcUri);
  var query = from genre in context.Genres
              where genre.Name == "Independent"
              from title in genre.Titles
              where title.ReleaseYear>=2007
              select title ;
  var titles = query.ToList();
  return View(titles);
}
```

Code download available at code.msdn.microsoft.com/mag201102DataPoints.

The markup in the HomeController Index view (\Views\HomeController\index.aspx) is where all of the interesting presentation logic is being performed. To leverage jQuery and the DataTables plug-in, you'll need to add a set of script files to your project. Alternatively, you can point to the online set of scripts (see the Microsoft AJAX Content Delivery Network at asp.net/ajaxLibrary/CDN.ashx), but I've chosen to host them locally. The download for the DataTables plug-in contains a \media folder (which contains the scripts) that you can drop into your project. You can see I've already done this in **Figure 1**.

Figure 2 contains the code listing of the Index.aspx file.

The CSS link and two script sources at the beginning of the <head> section point to CSS formatting and the critical jQuery and jQuery.datatables JavaScript files.

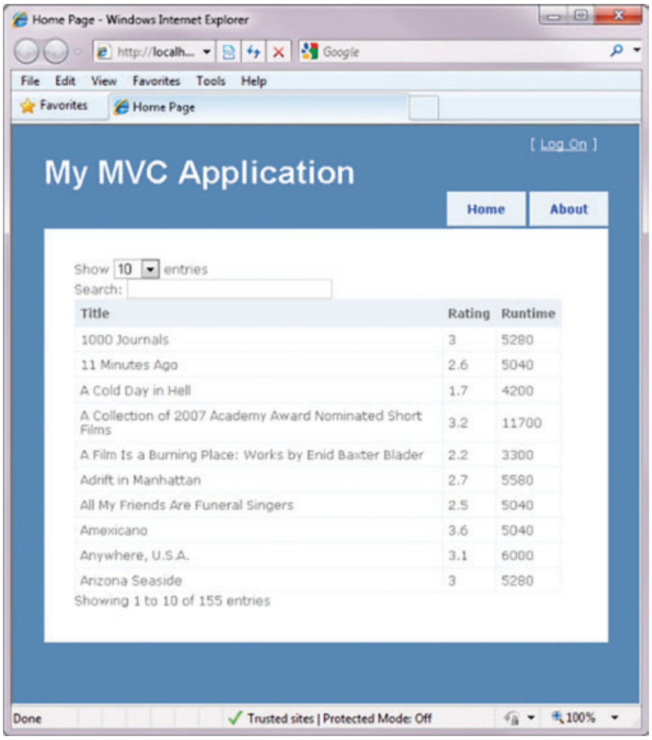
Next, let's focus on the table as it's laid out in the page. The DataTables plug-in is dependent on the table's ID and header information stored in the <thead> section. After this, a bit of code iterates through the IEnumerable<Title> passed in to the View from the HomeController.cs file and displays the Name, AverageRating and Runtime values in the appropriate columns.

When the page initially starts up, the JavaScript method in the header tag uses jQuery to locate the Netflix table in the form and applies the dataTable function to it. DataTables is highly configurable, but with this simple form of calling the dataTable function, the referenced table, Netflix, will acquire the DataTables default configuration. **Figure 3** shows the resulting page.

Figure 2 **The HomeController Index.aspx**

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
 Inherits="System.Web.Mvc.ViewPage<IEnumerable<Title>>" %>
<%@ Import Namespace="JQueryMVC.Controllers" %>
<%@ Import Namespace="JQueryMVC.NetflixOData" %>
<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent"
  runat="server">
    Home Page
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent"
  runat="server">
  <head>
    <link href="../../media/css/demo-table.css"
      rel="stylesheet" type="text/css" />
    <script src="../../media/js/jquery.js"
      type="text/javascript"></script>
    <script src="../../media/js/jquery.dataTables.js"
      type="text/javascript"></script>

    <script type="text/javascript" charset="utf-8">
        $(document).ready(function () {
           $('#Netflix').dataTable();
        });
    </script>
  </head>
  <div>
    <table id="Netflix">
      <thead><tr><th>Title</th>
                 <th>Rating</th>
                 <th>Runtime</th></tr></thead>
      <tbody>
        <% foreach (Title title in Model)
           { %>
             <tr><td><%= title.Name %> </td>
                 <td><%= title.AverageRating %></td>
                 <td><%= title.Runtime %></td></tr>
           <% } %>
      </tbody>
    </table>
  </div>
</asp:Content>
```



Show 10 entries
Search:

Title	Rating	Runtime
1000 Journals	3	5280
11 Minutes Ago	2.6	5040
A Cold Day in Hell	1.7	4200
A Collection of 2007 Academy Award Nominated Short Films	3.2	11700
A Film Is a Burning Place: Works by Enid Baxter Blader	2.2	3300
Adrift in Manhattan	2.7	5580
All My Friends Are Funeral Singers	2.5	5040
Amexicano	3.6	5040
Anywhere, U.S.A.	3.1	6000
Arizona Seaside	3	5280

Showing 1 to 10 of 155 entries

Figure 3 **Displaying Data with the DataTables Plug-In**

DataTables has done more than use CSS to make the table pretty. Notice that, at the bottom, it tells you that it retrieved 155 rows. By default it does client-side paging starting at 10 entries per page, although the user can choose 25, 50 or 100 entries per page in the drop-down. The Search box filters the results based on its findings in all of the available columns in the table. The user can also click on the header row columns to sort the data. The DataTables plug-in feature set is so rich that there are even plug-ins for the plug-in. You can learn much more about enhancing the default on the datatables.net Web site.

Querying OData on the Client Side

You don't always have the benefit of working with a client API, so I'll flip over to the more challenging task of querying the Netflix OData on the client side without the benefit of one of the other available libraries (the AJAX client library). I'll employ the DataTables plug-in while getting around some of the limitations posed by the Netflix service. You're likely to run into these same limitations when working with other public OData services as well.

> DataTables has done more than use CSS to make the table pretty.

For this round, I'm using an ASP.NET WebForms app, though I could be using plain old HTML because I'm not using any .NET Framework code in this page. You'll need the \media folder in this application as well, but you won't be creating a proxy to the service, so there's no need to use Add Service Reference.

Figure 4 **Preparing OData Results for Display**

```
function displayResults(results) {
  var entities;
  var redraw;

// Find data in results
  if (results.d[0] == undefined) {
    queryNext = results.d.__next;
    entities = results.d.results;
  }
  else {
    queryNext = "";
    entities = results.d;
  }

  // Instantiate dataTable if necessary
  if (oTable ==null)
    oTable = $('#Netflix').dataTable();

  // Build table rows from data using dataTables.Add
  for (var post in entities) {
    if (post == queryResults.length-1)
      redraw = true; //only redraw table on last item
    else
      redraw = false;

    oTable.fnAddData([
      entities[post].Name, entities[post].Rating,
      entities[post].Runtime],redraw);
  }

  // Continue retrieving results
  if (queryNext > "") {
    query = FixNetFlixUrl(queryNext);
    getData();
  }
}
```

The dataTable function has a method called sAjaxSource that will automatically retrieve data from the target source. But this requires your results to be formatted in a specific way. OData results don't align with this. There's a great blog post written by a California-based developer, Jeff Morris, that demonstrates reshaping OData results in a WCF Data Services query interceptor. You can read the post at bit.ly/bMPzTH.

Instead, I'll use AJAX to return the OData in its native form and then manually populate the table.

The body of the page starts out with the table and its <theader> defined (again, required by DataTables), as well as an empty <tbody>:

```
<body>
  <form id="form1" runat="server">
    <table id="Netflix" width="100%">
      <thead>
        <tr><th width="50%">Title</th>
            <th>Rating</th>
            <th>Runtime</th></tr>
      </thead>
      <tbody id="netflixBody"/>
    </table>
  </form>
</body>
```

The page has a number of functions: GetData, displayResults and a helper function to deal with one of the current shortcomings of the Netflix service. Similar to the .NET Client Library for OData, there's a client library for AJAX that's part of the Microsoft ASP.NET AJAX APIs. Here's an example from the AJAX documentation of what a JavaScript OData query looks like using this library:

```
function doQuery() {
var northwindService = new
Sys.Data.OpenDataServiceProxy("/Northwind.svc");
northwindService.query("/Customers", cbSuccess, cbFailure, userContext);
```

Alternatively, you can use straight AJAX and jQuery as I do in the following examples. Let's look at the beginning of the header script, including the getData function:

```
<script type="text/javascript" charset="utf-8">
  var oTable;
  var query = "http://odata.netflix.com/v1/Catalog/
Titles?$orderby=Name&$top=500"

  $(document).ready(function () { getData() });

  function getData() {
    var url = query + "&$callback= displayResults"
      + "&$format=json";
    $.ajax({ dataType: "jsonp", url: url });
  }
```

When the page begins, the document.ready function automatically calls getData. getData constructs a URL from the pre-defined OData query and appends parameters to return the OData as JSON (an alternative to the default AtomPub format), as well as defining the method to be executed when the AJAX call is complete.

When the AJAX call is complete, the displayResults function will be called using the results from the OData query (see **Figure 4**).

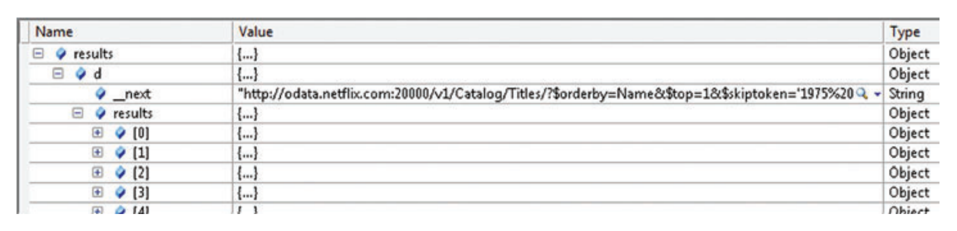
The section of code that's commented with "find data in results" is handling one of the Netflix limitations I've mentioned. Netflix is enforcing server-side paging to protect its servers and only returns 500 rows per request. Can you imagine if someone lazily queried for all movies? I'm sure that happens frequently. The server-side paging doesn't prevent you from getting additional rows; you just need to do this explicitly.

Handling large amounts of data in the client is exactly what DataTables is brilliant at, and there's a good chance you'll want to take advantage of it. It may take a bit longer to load all of the data when you're retrieving large quantities (for example, 5,000 rows), but once they're in memory, DataTables can let the end user do all types of filtering and sorting on that data.

When I first saw DataTables demonstrated, the person showing it off said they were using it for a corporate reporting tool where they were downloading 80,000 rows. I protested loudly to this abuse of the Internet and the server. However, having seen DataTables in action, I'm no longer so opposed to this usage in a controlled scenario.

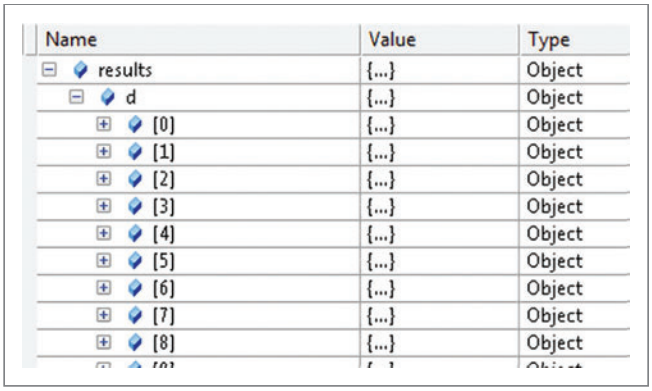
OData provides a way to easily request another batch of data and Netflix provides this hook for you to take advantage of. Here's a query that requests 501 results:

```
http://odata.netflix.com/v1/Catalog/Titles?$orderby=Name&$top=501
```



Name	Value	Type
⊟ results	{...}	Object
⊟ d	{...}	Object
__next	"http://odata.netflix.com:20000/v1/Catalog/Titles/?$orderby=Name&$top=1&$skiptoken='1975%20	String
⊟ results	{...}	Object
⊞ [0]	{...}	Object
⊞ [1]	{...}	Object
⊞ [2]	{...}	Object
⊞ [3]	{...}	Object
⊞ [4]	{...}	Object

Figure 5 **JSON Results of a Request for More Data than the Service Is Configured to Return**



Name	Value	Type
⊟ results	{...}	Object
⊟ d	{...}	Object
⊞ [0]	{...}	Object
⊞ [1]	{...}	Object
⊞ [2]	{...}	Object
⊞ [3]	{...}	Object
⊞ [4]	{...}	Object
⊞ [5]	{...}	Object
⊞ [6]	{...}	Object
⊞ [7]	{...}	Object
⊞ [8]	{...}	Object

Figure 6 **JSON Results for a Request Within the Configured Return Amount**

When the query exceeds that service's limit, Netflix uses the OData continuation token feature. In addition to the entries, the results contain one more element after the last entry. Here it is in AtomPub format:

```
<link rel="next"
  href="http://odata.netflix.com:20000/v1/Catalog/Titles/?$orderby=
Name&$top=1&$skiptoken='1975%20Oklahoma%20National%20Championship%20
Game','BVZUb'" />
</feed>
```

The skiptoken parameter tells the query where to start with the next set of results. In JSON, that entry is seen at the beginning of the results in a property called __next, as shown in **Figure 5**.

When a query doesn't exceed the limitation, the entries are directly inside the d property, as you can see in **Figure 6**. That's why GetData needs to test to see where it will find the results. If there's a continuation token, it stores that in NextQuery and then executes the continuation query in order to build up the complete result set in memory.

If you look at the __next property, you'll notice that Netflix added a port number, 20,000, to the query. However, if you execute that query directly, it will fail. So you'll need to remove the port number from the URI before requesting it. That's the purpose of the FixNetFlixUrl function that I call prior to making the request.

You'll have to watch for anomalies such as this when consuming public OData services. You've now seen how to deal with a service that limits the number of results it will return and one that inflicts a breaking change in its continuation token.

For each set of results that is retrieved, the method uses the DataTables fnAddData method to add each item to the table. Redrawing the table is expensive, so I've set the redraw parameter of fnAddData to false until reaching the last item in the results. Redrawing throughout the data retrieval makes the UI more fluid, rather than waiting until all 5,000 rows have been retrieved and added to the table.

After modifying the initial query to return 5,000 rows in my rural-Vermont-with-poky-Internet-access environment, and postponing the redraw to the bitter end, it took nearly a minute to capture all of the rows and display the table. Redrawing every row was much snappier and I was able to interact with the table even as more rows were being added. That was a nice surprise.

Once all 5,000 rows were in the table, DataTables did an amazing job of sorting and searching. Sorting took less than one second. Search was instantaneous, as it responds to each keystroke in the search box (see **Figure 7**).

A Small Adjustment for Internet Explorer 8

A recent update to DataTables triggers an Internet Explorer 8 feature that's not at all desirable when working with large result sets in DataTables. Internet Explorer displays a warning message when too many lines of script are being executed.

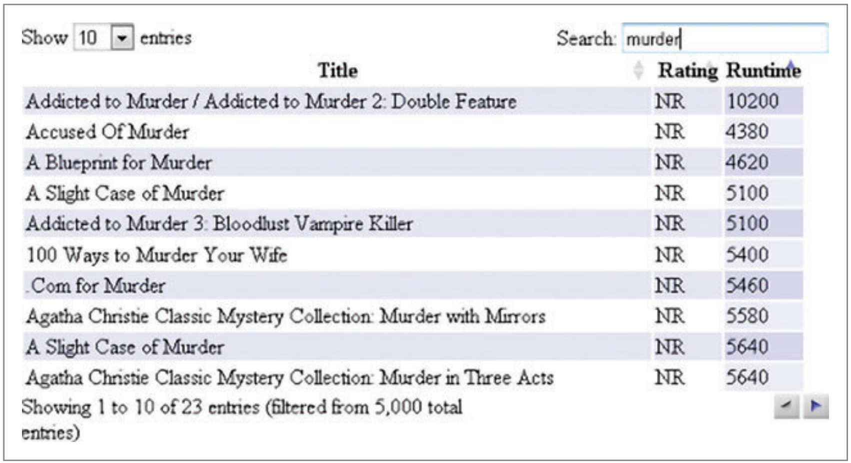
The Microsoft support site recommends adjusting the client machine's registry to change this behavior. That's not a reasonable solution for fixing this application; I don't want to mess with a client's registry settings if it can be avoided. But there's another option.

A post in the DataTables user forums suggested a modification to the DataTables script file. I implemented this modification and it works like a charm. You can see the details in the forum thread titled, "Sorting causes IE to throw 'A script on this page is causing Internet Explorer to run slowly,'" at bit.ly/co4AMD.

So Many Features to Explore

Hopefully you've already seen enough to understand my excitement over this extensive plug-in. There's a lot more that you can do to configure a table's look—as well as its behavior—in the read-only scenario I've demonstrated. DataTables also lets you edit in a fluid way, and if you want to keep some of that logic on the server side, you can do that while still benefiting from DataTables.

Using DataTables to let your end users slice and dice the large amounts of data available in the increasing number of publicly available OData services seems, to me, like a match made in geek heaven. ■



Show 10 entries Search: murder

Title	Rating	Runtime
Addicted to Murder / Addicted to Murder 2: Double Feature	NR	10200
Accused Of Murder	NR	4380
A Blueprint for Murder	NR	4620
A Slight Case of Murder	NR	5100
Addicted to Murder 3: Bloodlust Vampire Killer	NR	5100
100 Ways to Murder Your Wife	NR	5400
.Com for Murder	NR	5460
Agatha Christie Classic Mystery Collection: Murder with Mirrors	NR	5580
A Slight Case of Murder	NR	5640
Agatha Christie Classic Mystery Collection: Murder in Three Acts	NR	5640

Showing 1 to 10 of 23 entries (filtered from 5,000 total entries)

Figure 7 **Real-Time Search Results in DataTables**

JULIE LERMAN *is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other Microsoft .NET Framework topics at user groups and conferences around the world. Lerman blogs at thedatafarm.com/blog and is the author of the highly acclaimed book, "Programming Entity Framework" (O'Reilly Media, 2010). Follow her on Twitter.com: @julielerman.*

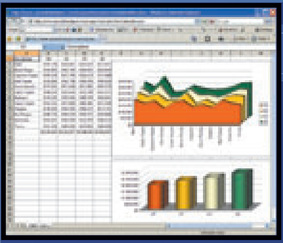
THANKS to the following technical experts for reviewing this article: *Rey Bango and Alex James*

In a League by Itself

"SpreadsheetGear 2010 is Fantastic! These new capabilities just propelled this control way-way past any competition, of which you have none IMHO. SpreadsheetGear is in a league all by itself."

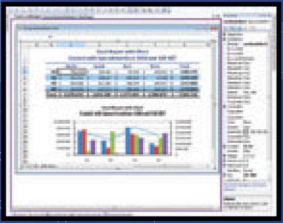
Greg Newman, Senior Software Engineer, WSFS Cash Connect

SpreadsheetGear 2010



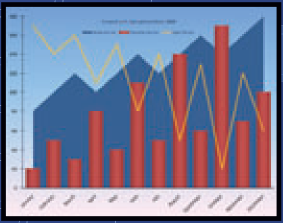
ASP.NET Excel Reporting

Easily create richly formatted Excel reports without Excel using the new generation of spreadsheet technology built from the ground up for scalability and reliability.



Excel Compatible Windows Forms Control

Add powerful Excel compatible viewing, editing, formatting, calculating, charting and printing to your Windows Forms applications with the easy to use WorkbookView control.



Create Dashboards from Excel Charts and Ranges

You and your users can design dashboards, reports, charts, and models in Excel rather than hard to learn developer tools and you can easily deploy them with one line of code.

Download the FREE fully functional 30-Day evaluation of SpreadsheetGear 2010 today at

www.SpreadsheetGear.com.

SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com

Branch-Node Synchronization with SQL Azure, Part 2: Service-Based Sync

Last month, I focused on the general architecture and design for synchronizing corporate databases with SQL Azure and with various destination nodes from there. I talked about optimizing by filtering and also using geographic distribution—or using the two strategies in conjunction—to optimize the overall data distribution and collection network.

This month, I'm going to pull in Windows Azure to host a synchronization service and focus on synchronizing via a service interface in the cloud. This will provide a means of scaling the synchronization mechanism to handle many more end nodes than could be handled by direct-to-database synchronization. I'll use the October 2010 Community Technology Preview (CTP) release of the Microsoft Sync Framework 4.0 (bit.ly/dpyMP8), which is built on top of the 2.1 framework used in January's issue.

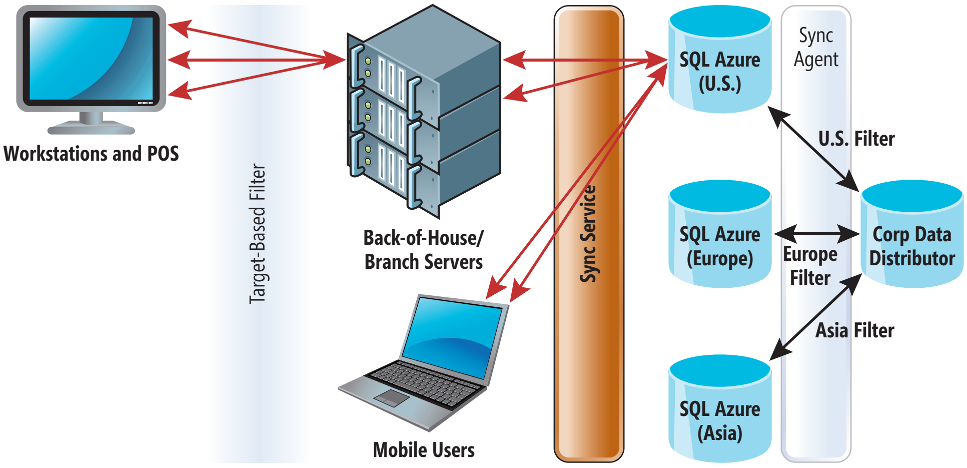


Figure 1 **Typical Corporate Architecture**

A synchronization service can be built straight on top of version 2.1 and a good sample and walk-through can be found at bit.ly/bibIdl and bit.ly/epyImQ. However, with the availability of the 4.0 CTP and the Internet-focused elements of the release, it makes good sense to leverage it for the Windows Azure Sync Service. A fair amount of code still needs to be written to make a functional solution, but in the end, we'll end up with a synchronization service that could be consumed by any device using OData.

Synchronizing at Internet Scale

I covered some ideas last month on how to scale the direct-to-database sync. However, in some—if not many—cases there are a number of reasons that the scale problem isn't so easily solved. Giving just some cursory thought to things that won't be covered by using the previously described approaches, one can easily come up with the following:

1. Due to the relationship of the data, it can't easily be split.
2. There isn't segmentation that makes sense and any split would be arbitrary, which would likely lead to unforeseen hot-spots in the various partitions of the solution.
3. The amount of data that would need to be replicated would drive up the cost beyond being cost-effective if it had to exist in multiple places.
4. Burst times for synchronization; for example, end-of-day processing for hundreds or thousands of retail locations creates contention regardless of the partitioning.

This obviously doesn't exhaust all of the possible reasons that would necessitate a design other than straight to SQL Azure for synchronization, but it's a good enough list to broach the topic and take a look at how to solve the problem. As with most things in computer science, I'll attempt to resolve the issues above by inserting a layer of indirection. In this case, it will be a service layer hosted in a Windows Azure Web Role used as the synchronization point instead of synchronizing directly with the SQL Azure instance. I've updated the end state diagram from last month by adding a placeholder for the Sync Service hosted in Windows Azure, arriving at a logical design such as is shown **Figure 1**.

Getting Started

The Sync Framework 4.0 is particularly suited to help solve this problem. However, it will require that I do a bit more work than the simple model of synchronizing directly between databases. The 4.0 CTP shipped with a good sample and walk-through in the help

This article discusses a prerelease version of the Sync Framework 4.0; all information is subject to change.

Weekly Schedule

Explanation of Benefits Report

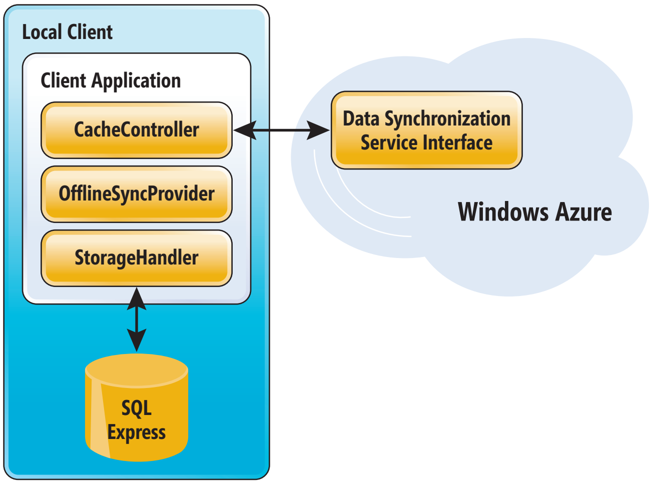


Figure 2 **Sync Framework 4.0 Client Synchronization Objects**

file, which is titled "Creating a Sync Service in Windows Azure." I'll use it as the basis for discussing the Sync Service implementation. The client code is a bit more difficult because in version 4.0 there isn't a client-side runtime library to assist, due to the abstraction created to open up synchronization to any platform that will work with OData. However, there's a sample in the 4.0 samples for Windows Mobile 6.5 using SQL Server CE. I've co-opted the code needed and modified it to work with standard SQL Server. To start, the October 2010 4.0 CTP uses a certain set of objects to perform the synchronization activity and it helps to be familiar with them. The client application consumes a CacheController, which is responsible for communication to the Sync Service using OData. On the local side, the CacheController uses an OfflineSyncProvider, which is the data-store-specific—and likely per-target-platform—interface between the application and the data (see **Figure 2**). In this implementation, being based on the sample, there's a StorageHandler object used to handle the local data access. The OfflineSyncProvider is a known type that's used by the CacheController, but the StorageHandler is custom code written to handle all of the back-end store interaction. Think of the OfflineSyncProvider as the intelligence on top of the data-access library and the StorageHandler as the data-access library. Notably, the 4.0 CTP only ships with a built-in CacheController for Isolated Storage in a Silverlight client, which leaves me with some work to do in order to use standard SQL Server. The layout of the objects and interaction boundaries are represented at a high level in **Figure 2**.

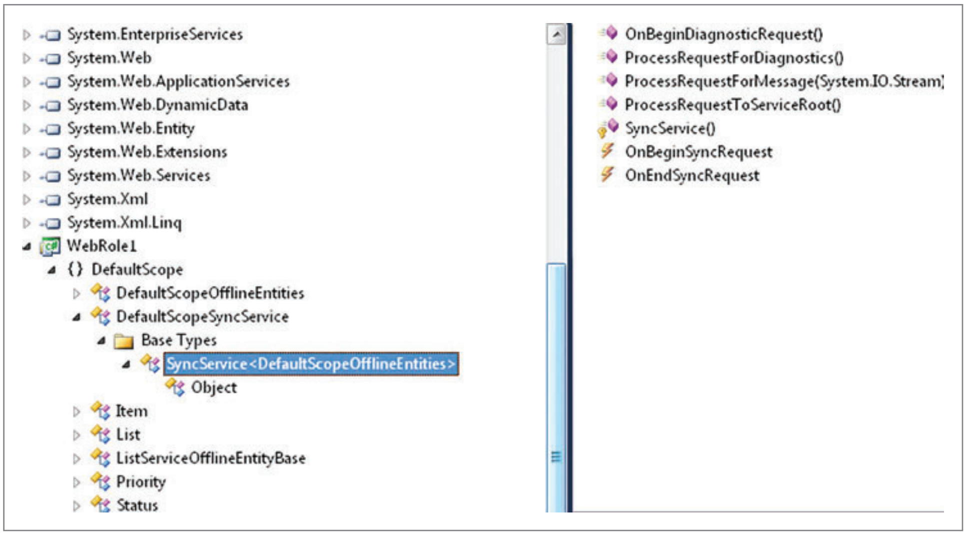
Developing the Cloud Sync Service

I was always told to deliver the bad news first and the good news last. That way, the conversation—and hopefully the spirits of those participating in it—ends on a positive note. However, in this case, I'm going to reverse the delivery order, hoping to sell the solution on the merits of the easy part. The bulk of the work comes on the client side, but the framework provides a lot of assistance for the server side. In a design session I lead, I was once told by someone that sold death-care services (funerals, plots, coffins and so on) that they'd never make a single sale if they focused on "what it is" and that instead the focus needed to be on "what it does"; in the case of death care, peace of mind was the real commodity being bought, not a coffin and hole in the ground. Such is the case with the Sync Framework. The Sync Framework 2.1 took care of a lot of things for the developer, but it fell a little short of goal when it came to service-based synchronization. It didn't at all address the plethora of devices and platforms that might want to synchronize with the data that's made available through an Internet-facing synchronization service. With the—now popularly termed—*consumerization* of IT, my customers find themselves having to deal with many devices in the hands of people at all levels of the enterprise. Sync Framework 4.0 CTP is aimed at helping with this type of challenge, particularly in regard to synchronizing data to those devices.

Getting the server side of this solution up and going is quite simple. Basically, it comes down to these steps:

1. Define the database
2. Create a configuration file for it
3. Use the SyncServiceUtil to provision the database using the config file
4. Use the SyncServiceUtil to generate classes required for the synchronization service
5. Create a Windows Azure-based Web Role to host the service
6. Deploy

If you're like me when you read this summary, you think, "what configuration file?" The schema for the file can be found in the MSDN Library at bit.ly/h2FJod. Using that and referencing the ListDB database and the related config file for it that ships with the 4.0 samples, one can put together a custom config file that represents a database with



```
▷ System.EnterpriseServices
▷ System.Web
▷ System.Web.ApplicationServices
▷ System.Web.DynamicData
▷ System.Web.Entity
▷ System.Web.Extensions
▷ System.Web.Services
▷ System.Xml
▷ System.Xml.Linq
◢ WebRole1
  ◢ {} DefaultScope
    ▷ DefaultScopeOfflineEntities
    ◢ DefaultScopeSyncService
      ◢ Base Types
        ◢ SyncService<DefaultScopeOfflineEntities>
            Object
    ▷ Item
    ▷ List
    ▷ ListServiceOfflineEntityBase
    ▷ Priority
    ▷ Status
```

```
OnBeginDiagnosticRequest()
ProcessRequestForDiagnostics()
ProcessRequestForMessage(System.IO.Stream)
ProcessRequestToServiceRoot()
SyncService()
OnBeginSyncRequest
OnEndSyncRequest
```

Figure 3 **Object Browser View of SyncServices Generated Code**

minimal confusion. Once this file exists, the creation of the Windows Azure-based services is a snap. First, the target database—in this case the ListDB sample in the 4.0 SDK—needs to be created in Windows Azure. Once that's done, the new SyncServiceUtil can be used to provision the database using a command similar to:

```
SyncSvcUtil /mode:provision
/scopeconfig:listdbconfig.xml
```

The one thing that will have to be set in the config file is the connection to the SQL Azure database. Near the end of the config file is the <TargetDatabase /> element, which will need to be configured properly for the cloud:

```
<Databases>
  <TargetDatabase Name="listdb" DbServer="[URI for the SQL Azure DB
   Instance]" DbName="listdb" UserName="[username]" Password="[password]"
   UseIntegratedAuth="false" />
</Databases>
```

Running the utility will generate two files: DefaultScopeEntities.cs and DefaultScopeSyncServices.svc. The "DefaultScope" part of the name comes from the config file and is found in the element <SyncScope />:

```
<SyncScope Name="DefaultScope" IsTemplateScope="true">
```

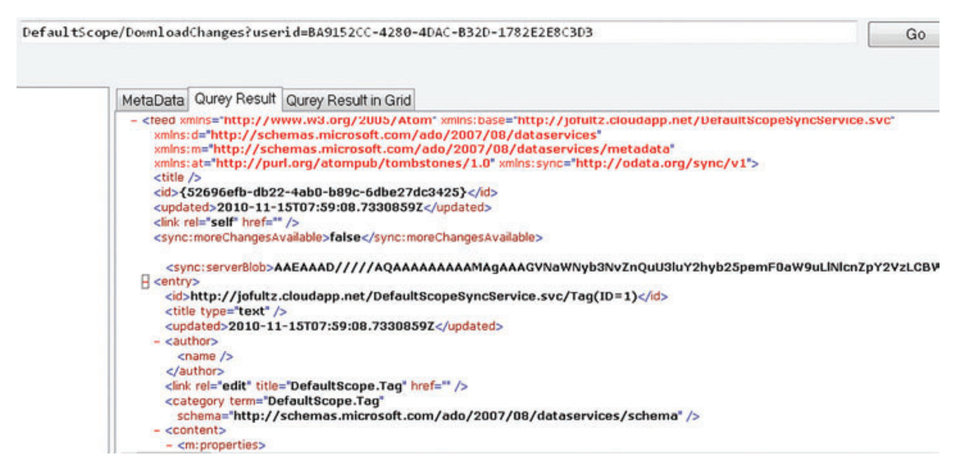
The entities file is pretty much as described, but the DefaultScopeSyncServices.svc file is somewhat more significant as it generates the partial class that allows me to intercept service calls and add custom logic (something new to 4.0). The base synchronization logic is all included as part of the base object. **Figure 3** shows the DefaultScopeSyncService class and the related entities class as the template type for the template class SyncService.

Note, on the right-hand side of **Figure 3**, the abbreviated list of service interfaces that are exposed to do the synchronizing (as compared to what would need to be exposed using Sync Framework 2.1 directly). If I wanted to add any custom logic to the synchronization process, I'd simply open the DefaultScopeSyncServices.svc file, pick the method interceptor and write to my heart's content. To implement basic synchronization via the service interface that was just created, I simply need to associate the service/Web project containing the files with a Web Role and in the WebRole:OnStart method add a line to create the activation context:

```
public override bool OnStart()
{
  DiagnosticMonitor.Start("DiagnosticsConnectionString");

  // For information on handling
  // configuration changes, see the MSDN topic at
  // go.microsoft.com/fwlink/?LinkId=166357
  RoleEnvironment.Changing += RoleEnvironmentChanging;
  Microsoft.Samples.Synchronization.ActivationContext.
    CreateActivationContext();
  return base.OnStart();
}
```

I then make a couple of configuration changes to ensure that the Sync Framework binaries are set to CopyAlways. To get the new service interface goodness, I ensure that the 4.0 Microsoft.Synchronization.dll is both referenced and set to be published with the package. Then I publish it to my Web Role and I'm ready to go.



```
DefaultScope/DownloadChanges?userid=BA9152CC-4280-4DAC-B32D-1782E2E8C3D3
```

```
- <feed xmlns="http://www.w3.org/2005/Atom" xmlns:base="http://jofultz.cloudapp.net/DefaultScopeSyncService.svc"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns:at="http://purl.org/atompub/tombstones/1.0" xmlns:sync="http://odata.org/sync/v1">
    <title />
    <id>{52696efb-db22-4ab0-b89c-6dbe27dc3425}</id>
    <updated>2010-11-15T07:59:08.7330859Z</updated>
    <link rel="self" href="" />
    <sync:moreChangesAvailable>false</sync:moreChangesAvailable>

      <sync:serverBlob>AAEAAAD/////AQAAAAAAAAAMAgAAAGVNaWNyb3NvZnQuU3luY2hyb25pemF0aW9uLlNlcnZpY2VzLCBV
  - <entry>
      <id>http://jofultz.cloudapp.net/DefaultScopeSyncService.svc/Tag(ID=1)</id>
      <title type="text" />
      <updated>2010-11-15T07:59:08.7330859Z</updated>
    - <author>
        <name />
      </author>
      <link rel="edit" title="DefaultScope.Tag" href="" />
      <category term="DefaultScope.Tag"
        schema="http://schemas.microsoft.com/ado/2007/08/dataservices/schema" />
    - <content>
      - <m:properties>
```

Figure 4 **OData Viewer Tool DownloadChanges Result**

I can make a simple test by requesting the sync scopes that are currently available by entering a request such as jofultz.cloudapp.net/defaultscopeSyncService.svc/$syncscopes into my browser. I get back the following response, which gives me some confidence that the service is working:

```
- <service xml:base="http://rd00155d3a1a55:20000/
defaultscopesyncservice.svc/" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns="http://www.w3.org/2007/app">
- <workspace>
  <atom:title>SyncScopes</atom:title>
- <collection href="defaultscope">
  <atom:title>defaultscope</atom:title>
  </collection>
  </workspace>
  </service>
```

I could also request other data, and if there are changes, I'll get them as OData by default. I can do this in a browser or via a tool. Using the OData Viewer Tool on CodePlex (dataservicestool.codeplex.com/releases/view/52805), I issue the request to download changes: jofultz.cloudapp.net/defaultscopeSyncService.svc/DefaultScope/DownloadChanges?userid=BA9152CC-4280-4DAC-B32D-1782E2E8C3D3, which gives me the results as shown in **Figure 4**.

The great news here is that the additions in Sync Framework 4.0 CTP provide the simplified synchronization interfaces with results retrievable in OData ATOM and OData JSON. This opens up the synchronization to other platforms from the client standpoint and relegates proprietary data formats to being legacy formats—and all I had to do was run a utility, configure a project and add a line of code.

Figure 5 **Segmentation of Work for Client and Service**

Unit of Work	Effort
DB Schema for Sync (Server)	Configuration
Service Implementation	Generated + 1 line of code
Customize Validation Hooks in Sync	Hooks Generated; Only Have to Write Value-Add Code
DB Schema for Sync (client)	Could Use 2.1 Provisioning or Custom
Sync Implementation for Non-Silverlight	Custom
Sync Client for Silverlight	Configuration + Generation

INFRAAIR
Select class
All
First
Global
Travel+
Travel
Selected seat
SEATMAP KEY
CUSTOMER SATISFACTION
FLIGHT AMENITIES
Food
Audio
Video
Power
Internet
Infant
CLASS
SEATING DETAILS
PITCH
WIDTH

Implementing Client Synchronization

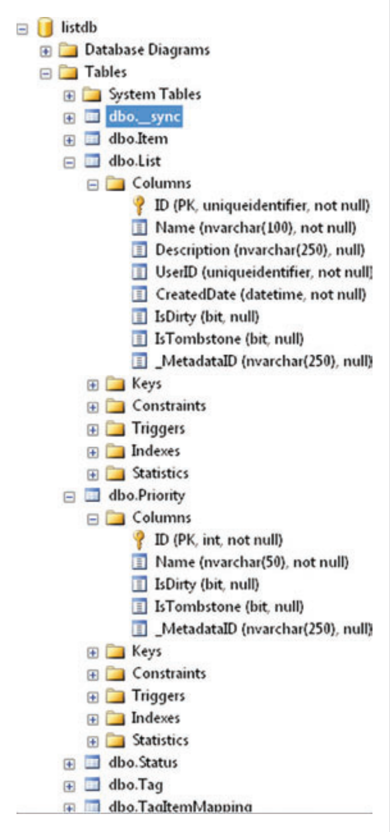
Here's the point in the implementation where the headphones go on and multitasking gives way to focus. The cloud service was almost a matter of configuration, but the client piece takes a bit more work if you're starting from scratch. Because the Sync Framework 4.0 CTP ships with a CacheController for isolated storage, if Silverlight is the target client platform, then the client implementation will be as easy as the cloud service implementation. However, my target is a Windows client running SQL Server Standard/Express, and that will require some effort. The SyncServiceUtil still assists by generating the needed entities, but a custom CacheController and OfflineSyncProvider will have to be created. More importantly, the data store will need to be modified in order to facilitate the change tracking. One might do this with a version 2.1-provisioned database or his own custom schema for change tracking. Such an implementation could add significant work and complexity to the overall application in terms of a more complicated database implementation and a more complicated code base. However, it needs to be done in order to leverage the rest of the framework. In describing this to others, I get asked, "Why not just do it all yourself?" The answer is simple: I do it this way to reduce the body of work and open the implementation to synchronization by other 2.1 and 4.0 framework sync clients/agents (including non-Windows platforms).

Take a look at the segmentation of work shown in **Figure 5** for just the client and service pieces being discussed. You can see that using the framework reduces the amount of work by roughly 60 percent or more depending on the target client platform.

Working with the Mobile 6.5 and SQL CE sample provides me with an example implementation of what you might do with the database in order to implement the client synchronization; note the IsDirty, IsTombstone and Metadata fields as seen in **Figure 6**.

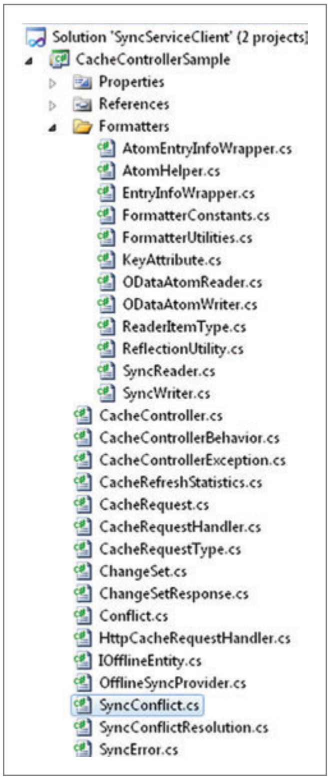
With a schema in place, I need a number of other things:

1. CacheController implementation as mentioned before
   a. Local store interaction
   b. Service interaction
   c. Sync Conflict Handler
   d. Sync Error Handler
2. Some code to generate and consume the OData
3. Code definition for the entities being synchronized
4. OfflineSyncProvider for the local SQL Server database



```
listdb
  Database Diagrams
  Tables
    System Tables
    dbo._sync
    dbo.Item
    dbo.List
      Columns
        ID (PK, uniqueidentifier, not null)
        Name (nvarchar(100), not null)
        Description (nvarchar(250), null)
        UserID (uniqueidentifier, not null)
        CreatedDate (datetime, not null)
        IsDirty (bit, null)
        IsTombstone (bit, null)
        _MetadataID (nvarchar(250), null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
    dbo.Priority
      Columns
        ID (PK, int, not null)
        Name (nvarchar(50), not null)
        IsDirty (bit, null)
        IsTombstone (bit, null)
        _MetadataID (nvarchar(250), null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
    dbo.Status
    dbo.Tag
    dbo.TagItemMapping
```

Figure 6 **Columns to Support Custom Synchronization Implementation**



```
Solution 'SyncServiceClient' (2 projects)
  CacheControllerSample
    Properties
    References
    Formatters
      AtomEntryInfoWrapper.cs
      AtomHelper.cs
      EntryInfoWrapper.cs
      FormatterConstants.cs
      FormatterUtilities.cs
      KeyAttribute.cs
      ODataAtomReader.cs
      ODataAtomWriter.cs
      ReaderItemType.cs
      ReflectionUtility.cs
      SyncReader.cs
      SyncWriter.cs
    CacheController.cs
    CacheControllerBehavior.cs
    CacheControllerException.cs
    CacheRefreshStatistics.cs
    CacheRequest.cs
    CacheRequestHandler.cs
    CacheRequestType.cs
    ChangeSet.cs
    ChangeSetResponse.cs
    Conflict.cs
    HttpCacheRequestHandler.cs
    IOfflineEntity.cs
    OfflineSyncProvider.cs
    SyncConflict.cs
    SyncConflictResolution.cs
    SyncError.cs
```

Figure 7 **Files Used from the 6.5 Sample**

For items 1 and 2, I use the code provided for the 6.5 sample (see **Figure 7**) and place it in my own CacheController project, which consists entirely of code borrowed from the sample.

I use the SyncServiceUtil to generate the entities using the same config file as before, along with the "/mode:codegen" and "/target:client" flags. This generates a DefaultScopeEntities.cs file that has my client-side objects. Because I'm stealing from the 6.5 sample, I copy settings.cs, utility.cs, SqlCeOfflineSyncProvider.cs, DataStoreHelper.cs and SqlCeStorageHandler.cs to my Windows Forms project. To minimize my coding effort I make the changes shown in **Figure 8**.

By leveraging the sample code and making those changes, I'm able to write a small console app that will call the Utility.Sync function, which in turn instantiates the OfflineSyncProvider and CacheController to perform the synchronization:

```
var localProvider = new
  SqlCeOfflineSyncProvider();
var controller = new CacheController(new
  Uri(Settings.SyncServiceUrl), Settings.
  SyncScope, localProvider);
```

So, one might ask, where's the code to do things like fetch changed records from the local store? All of that lives in the StorageHandler implementation. Take a look at **Figure 9** to see a piece of it.

Thus, the forward chain of operations works as follows:

1. Client app calls arbitrary sync function
2. Sync function
   a. Instantiates OfflineSyncProvider
   b. Instantiates CacheController (this one is custom), passing the service URI and the OfflineSyncProvider
   c. Finally calls CacheController.Refresh()
3. CacheController creates a CacheRequestHandler that will handle the communication with the Sync Service in Windows Azure
4. CachController asks the OfflineSyncProvider for the local changeset
5. The OfflineSyncProvider uses the StorageHandler to retrieve the changes from the local SQL Server

DEVELOP

Rich Business Intelligence Applications in WPF and Silverlight

Robust Pivot Grids for WPF and Silverlight let your users analyze data to make key business decisions. Visit infragistics.com to try it today!

NetAdvantage®
for Silverlight Data Visualization

NetAdvantage®
for WPF Data Visualization

Infragistics®

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91 80 4151 8042
@infragistics

Figure 8 **Changes to Sample Code Made to Minimize Coding Effort**

File / Project	Change
DefaultScopeEntities.cs	Rename the class to SqlCeOfflineEntity in order to match the expected type name in the borrowed files

	Add
	`[Microsoft.Samples.Synchronization.ClientServices.KeyAttribute]`
	in each place where
	`[System.ComponentModel.DataAnnotations.KeyAttribute()]`
	exists as it's used within the CacheController implementation

My new CacheController Project	Replace all namespaces with `namespace Microsoft.Samples.Synchronization.ClientServices.Custom`

SqlCeOfflineSyncProvider.cs	Replace
	`using Microsoft.Samples.Synchronization.ClientServices;`
	with
	`using Microsoft.Samples.Synchronization.ClientServices.Custom;`
	to reference my custom CacheController implementation

SqlCeStorageHandler.cs	Comment out all [connection].[transaction commands] from the file: working against SQL Server requires a little different implementation than SQL CE, and this would need to be added back properly for a real implementation
DataStoreHelper.cs	Change the connection string to point to my local SQL Server instance
Settings.cs	Assign SyncServiceUrl the URI for my Windows Azure Sync Service (http://jofultz.cloudapp.net/DefaultScopeSyncService.svc/)

Utility.cs	Replace
	`using Microsoft.Samples.Synchronization.ClientServices;`
	with
	`using Microsoft.Samples.Synchronization.ClientServices.Custom;`
	to reference my custom CacheController implementation

Figure 9 **Local Store Data Commands**

```
internal class SqlCeStorageHandler : IDisposable
  {
    #region SQL CE Commands

    private const string GET_ALL_PRIORITY = "SELECT [ID], [Name], [_
      MetadataID] FROM [Priority] WHERE [IsTombstone] = 0";

    private const string GET_ALL_STATUS = "SELECT [ID], [Name], [_
      MetadataID] FROM [Status] WHERE [IsTombstone] = 0";

    private const string GET_ALL_TAGS = "SELECT [ID], [Name], [_
      MetadataID] FROM [Tag] WHERE [IsTombstone] = 0";

    private const string GET_ALL_LISTS =
      "SELECT [ID], [Name], [Description], [UserID], [CreatedDate],
      [IsTombstone], [_MetadataID] FROM [List] WHERE [IsTombstone] = 0";

    private const string GET_ALL_ITEMS =
      "SELECT ID, ListID, UserID, Name, Description, Priority, Status,
      StartDate, EndDate, IsTombstone, [_MetadataID] FROM [Item] WHERE
      [IsTombstone]=0 AND [ListID]=@ListID";

    private const string SELECT_ITEM_CHANGES =
      "SELECT ID, ListID, UserID, Name, Description, Priority, Status,
      StartDate, EndDate, IsTombstone, [_MetadataID] FROM [Item] WHERE
      IsDirty = 1";

    private const string SELECT_LIST_CHANGES =
      "SELECT ID, Name, Description, UserID, CreatedDate, IsTombstone,
      [_MetadataID] FROM [List] WHERE IsDirty = 1";

    private const string SELECT_TAGITEMMAPPING_CHANGES =
      "SELECT TagID, ItemID, UserID, IsTombstone, [_MetadataID] FROM
      [TagItemMapping] WHERE IsDirty = 1";
```

6. CacheController uses the changeset to create a request and pass it to the CacheRequestHandler
7. The CacheRequestHandler uses the appropriate formatter (OData ATOM here) to create a proper request and sends it to the Sync Service URI

Of course, all of the unpacking and getting the data back to the client pretty much is just the same stuff in reverse. **Figure 4** shows the OData package as it flows back from the service.

Final Thoughts

Obviously, removing transaction support and keeping misnomers such as SqlCe[suffix] for objects isn't the way to go for any real implementation, but it served its purpose here to get a client version working without writing all-new code. Anyone wanting to create a SQL Server CacheController could easily start with the 6.5 sample and refactor and rename, with the major work coming in the commands inside of the StorageHandler that would need to be specific to their data store.

My primary goal here was to demonstrate a service-based synchronization architecture. I purposefully neglected caching and other optimizations that would need to happen for it to scale, but that's typically well understood. Also, I wanted to convey what's there, what isn't and what's possible while familiarizing the reader with the Sync Framework 4.0 CTP. I hope I've accomplished these things.

With the SQL Azure Data Sync CTP 2 underway, there's the promise of being able to set all of this up—including the client piece—via configuration and the download of a client-side agent. Of course, that would be for Windows-based machines, but if the goal was to reach a broader set of platforms, using the Sync Framework 4.0 directly might be the better choice.

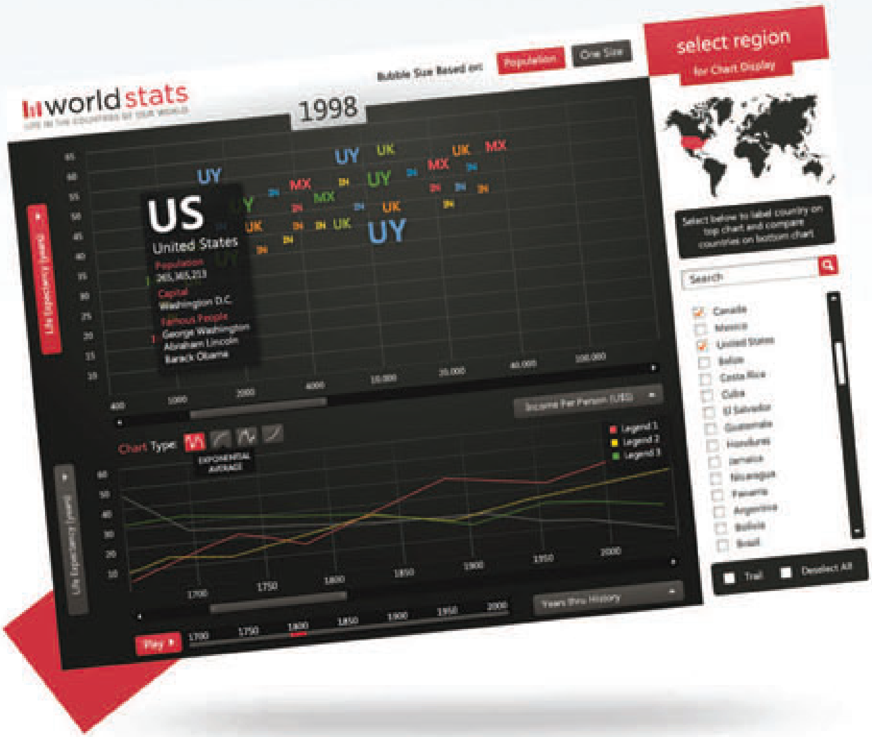
I want to encourage you to go out and download the latest Sync Framework SDK and at least follow the tutorial to set up the sync service in Windows Azure using a SQL Azure database and follow the example for the Silverlight Client to get a feel for it. For those who are a little braver, grab the files as described from the Windows Mobile 6.5 sample in the 4.0 CTP (there are two projects) and use them to create your own Windows-based synchronization client. ■

JOSEPH FULTZ *is an architect at the Microsoft Technology Center in Dallas, where he works with both enterprise customers and ISVs designing and prototyping software solutions to meet business and market demands. He's spoken at events such as Tech·Ed and similar internal training events.*

THANKS to the following technical expert for reviewing this article: *Ganeshan Iyer*

EXPERIENCE

Beautiful Data Visualizations That Bring Your Data to Life



Use our Motion Framework™ to see your data over time and give your users new insight into their data. Visit infragistics.com/motion to try it today!

NetAdvantage® ULTIMATE

for ASP.NET, Windows Forms, WPF, Silverlight,
WPF Data Visualization, Silverlight Data Visualization

Infragistics®

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91 80 4151 8042
@infragistics

Date	Country	Product
Wed Sep 1 2009 2:09:35 AM	US	🟦 Product A
Wed Sep 1 2009 6:03:14 AM	US	🟥 Product B
Wed Sep 1 2009 9:31:23 AM	Canada	🟨 Product C
Wed Sep 1 2009 10:56:54 AM	US	🟨 Product C
Wed Sep 1 2009 11:51:18 AM	US	🟨 Product C
Wed Sep 1 2009 1:20:20 PM	US	🟥 Product B
Wed Sep 1 2009 2:29:44 PM	France	🟦 Product A

Microsoft Silverlight

Microsoft Visual Studio

Week
Month
Quarter
Year
957
20%
189
30%
62
25%
16
Purchases
Profit Factor
Long Term Trending

Understanding the Dynamic Keyword in C# 4

Alexandra Rusina

The dynamic keyword and the Dynamic Language Runtime (DLR) are major new features in C# 4 and the Microsoft .NET Framework 4. These features generated a lot of interest when announced—along with a lot of questions. There were a number of answers as well, but they're now spread throughout documentation and on various technical blogs and articles. So people continue asking the same questions again and again on forums and at conferences.

This article provides a general overview of the new dynamic features in C# 4 and also delves into some more in-depth information about how they work with other language and framework features, such as reflection or implicitly typed variables. Given there's a lot of information available already, I'll sometimes reuse classic examples with links to the original sources. I'll also provide plenty of links for further reading.

This article discusses:

- Dynamic and type checking
- Dynamic versus object and var
- Using the DLR
- Working with dynamic objects

Technologies discussed:

Microsoft .NET Framework 4, C# 4, Microsoft Office

What Is Dynamic?

Programming languages are sometimes divided into statically typed and dynamically typed languages. C# and Java are often considered examples of statically typed languages, while Python, Ruby and JavaScript are examples of dynamically typed languages.

Generally speaking, dynamic languages don't perform compile-time type checks and identify the type of objects at run time only. This approach has its pros and cons: Often the code is much faster and easier to write, but at the same time you don't get compiler errors and have to use unit testing and other techniques to ensure the correct behavior of your application.

Originally, C# was created as a purely static language, but with C# 4, dynamic elements have been added to improve interoperability with dynamic languages and frameworks. The C# team considered several design options, but finally settled on adding a new keyword to support these features: dynamic.

The dynamic keyword acts as a static type declaration in the C# type system. This way C# got the dynamic features and at the same time remained a statically typed language. Why and how this decision was made is explained in the presentation "Dynamic Binding in C# 4" by Mads Torgersen at PDC09 (microsoftpdc.com/2009/FT31). Among other things, it was decided that dynamic objects should be first-class citizens of the C# language, so there's no option to switch dynamic features on or off, and nothing similar to the Option Strict On/Off in Visual Basic was added to C#.

Figure 1 **DynamicString**

```
public class DynamicString : DynamicObject {
  string str;

  public DynamicString(string str) {
    this.str = str;
  }

  public override bool TryInvokeMember(
    InvokeMemberBinder binder, object[] args,
    out object result) {

    Console.WriteLine("Calling method: {0}", binder.Name);

    try {
      result = typeof(string).InvokeMember(
        binder.Name,
        BindingFlags.InvokeMethod |
        BindingFlags.Public |
        BindingFlags.Instance,
        null, str, args);
      return true;
    }
    catch {
      result = null;
      return false;
    }
  }
}
```

When you use the dynamic keyword you tell the compiler to turn off compile-time checking. There are plenty of examples on the Web and in the MSDN documentation (msdn.microsoft.com/library/dd264736) on how to use this keyword. A common example looks like this:

```
dynamic d = "test";
Console.WriteLine(d.GetType());
// Prints "System.String".

d = 100;
Console.WriteLine(d.GetType());
// Prints "System.Int32".
```

As you can see, it's possible to assign objects of different types to a variable declared as dynamic. The code compiles and the type of object is identified at run time. However, this code compiles as well, but throws an exception at run time:

```
dynamic d = "test";

// The following line throws an exception at run time.
d++;
```

The reason is the same: The compiler doesn't know the runtime type of the object and therefore can't tell you that the increment operation is not supported in this case.

Absence of compile-time type checking leads to the absence of IntelliSense as well. Because the C# compiler doesn't know the type of the object, it can't enumerate its properties and methods. This problem might be solved with additional type inference, as is done in the IronPython tools for Visual Studio, but for now C# doesn't provide it.

However, in many scenarios that might benefit from the dynamic features, IntelliSense wasn't available anyway because the code used string literals. This issue is discussed in more detail later in this article.

Dynamic, Object or Var?

So what's the real difference between dynamic, object and var, and when should you use them? Here are short definitions of each keyword and some examples.

The object –keyword represents the System.Object type, which is the root type in the C# class hierarchy. This keyword is often used when there's no way to identify the object type at compile time, which often happens in various interoperability scenarios.

You need to use explicit casts to convert a variable declared as object to a specific type:

```
object objExample = 10;
Console.WriteLine(objExample.GetType());
```

This obviously prints System.Int32. However, the static type is System.Object, so you need an explicit cast here:

```
objExample = (int)objExample + 10;
```

You can assign values of different types because they all inherit from System.Object:

```
objExample = "test";
```

The var keyword, since C# 3.0, is used for implicitly typed local variables and for anonymous types. This keyword is often used with LINQ. When a variable is declared by using the var keyword, the variable's type is inferred from the initialization string at compile time. The type of the variable can't be changed at run time. If the compiler can't infer the type, it produces a compilation error:

```
var varExample = 10;
Console.WriteLine(varExample.GetType());
```

This prints System.Int32, and it's the same as the static type.

In the following example, no cast is required because varExample's static typed is System.Int32:

```
varExample = varExample + 10;
```

This line doesn't compile because you can only assign integers to varExample:

```
varExample = "test";
```

The dynamic keyword, introduced in C# 4, makes certain scenarios that traditionally relied on the object keyword easier to write and maintain. In fact, the dynamic type uses the System.Object type under the hood, but unlike object it doesn't require explicit cast operations at compile time, because it identifies the type at run time only:

```
dynamic dynamicExample = 10;
Console.WriteLine(dynamicExample.GetType());
```

This prints System.Int32.

> Generally speaking, dynamic languages don't perform compile-time type checks.

In the following line, no cast is required, because the type is identified at run time only:

```
dynamicExample = dynamicExample + 10;
```

You can assign values of different types to dynamicExample:

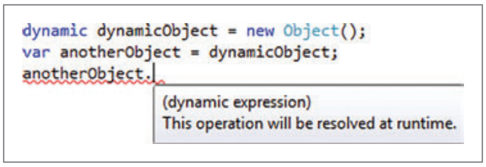
```
dynamicExample = "test";
```

There's a detailed blog post about differences between the object and dynamic keywords on the C# FAQ blog (bit.ly/c95hpl).

What sometimes causes confusion is that all of these keywords can be used together—they're not mutually exclusive. For example, let's take a look at this code:

```
dynamic dynamicObject = new Object();
var anotherObject = dynamicObject;
```

What's the type of anotherObject? The answer is: dynamic. Remember that dynamic is in fact a static type in the C# type system, so the compiler infers this type for the anotherObject. It's important to understand that the var keyword is just an instruction for the compiler to infer the type from the variable's initialization expression; var is not a type.



```
dynamic dynamicObject = new Object();
var anotherObject = dynamicObject;
anotherObject.
```

(dynamic expression)
This operation will be resolved at runtime.

Figure 2 **Dynamic Object in Visual Studio**

The Dynamic Language Runtime

When you hear the term "dynamic" in regard to the C# language, it usually refers to one of two concepts: the dynamic keyword in C# 4 or the DLR. Although these two concepts are related, it's important to understand the difference as well.

The DLR serves two main goals. First, it enables interoperation between dynamic languages and the .NET Framework. Second, it brings dynamic behavior to C# and Visual Basic.

The DLR was created based on lessons learned while building IronPython (ironpython.net), which was the first dynamic language implemented on the .NET Framework. While working on IronPython, the team found out that they could reuse their implementation for more than one language, so they created a common underlying platform for .NET dynamic languages. Like IronPython, the DLR became an open source project and its source code is now available at dlr.codeplex.com.

Another example where dynamic can help is creating dynamic method bags.

Later the DLR was also included in the .NET Framework 4 to support dynamic features in C# and Visual Basic. If you only need the dynamic keyword in C# 4, you can simply use the .NET Framework and in most cases it will handle all interactions with the DLR on its own. But if you want to implement or port a new dynamic language to .NET, you may benefit from the extra helper classes in the open source project, which has more features and services for language implementers.

Using Dynamic in a Statically Typed Language

It's not expected that everybody should use dynamic whenever possible instead of the static type declarations. Compile-time checking is a powerful instrument and the more benefits you can get from it, the better. And once again, dynamic objects in C# do not support IntelliSense, which might have a certain impact on overall productivity.

At the same time, there are scenarios that were hard to implement in C# prior to the dynamic keyword and the DLR. In most cases they used System.Object type and explicit casting and couldn't get much benefit from compile-time checking and IntelliSense anyway. Here are some examples.

The most notorious scenario is when you have to use the object keyword for interoperability with other languages or frameworks. Usually you have to rely on reflection to get the type of the object and to access its properties and methods. The syntax is sometimes hard to read and consequently the code is hard to maintain. Using dynamic here might be much easier and more convenient than reflection.

Anders Hejlsberg gave a great example at PDC08 (channel9.msdn.com/pdc2008/TL16) that looks like this:

```
object calc = GetCalculator();
Type calcType = calc.GetType();
object res = calcType.InvokeMember(
  "Add", BindingFlags.InvokeMethod,
  null, new object[] { 10, 20 });
int sum = Convert.ToInt32(res);
```

The function returns a calculator, but the system doesn't know the exact type of this calculator object at compile time. The only thing the code relies on is that this object should have the Add method. Note that you don't get IntelliSense for this method because you supply its name as a string literal.

With the dynamic keyword, this code looks as simple as this one:

```
dynamic calc = GetCalculator();
int sum = calc.Add(10, 20);
```

The assumptions are the same: There's some object with an unknown type that we expect to have the Add method. And similar to the previous example, you don't get IntelliSense for this method. But the syntax is much easier to read and use and looks similar to calling a typical .NET method.

Dynamic Method Bags

Another example where dynamic can help is creating dynamic method bags, which are objects that can add and remove properties and methods at run time.

The .NET Framework 4 has a new namespace: System.Dynamic. This namespace is in fact a part of the DLR. The System.Dynamic.ExpandoObject and System.Expando.DynamicObject classes in combination with the new dynamic keyword can help you to create dynamic structures and hierarchies in a clear and easy-to-read way.

For example, here's how you can add a property and a method by using the ExpandoObject class:

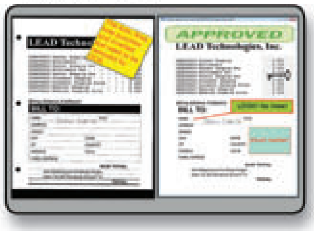
```
dynamic expando = new ExpandoObject();
expando.SampleProperty =
  "This property was added at run time";
expando.SampleMethod = (Action)(
  () => Console.WriteLine(expando.SampleProperty));
expando.SampleMethod();
```

For more in-depth scenarios, be sure to take a look at the MSDN documentation for the ExpandoObject and DynamicObject classes. Also, it's worth reading through the articles "Dynamic Method Bags" by Bill Wagner (msdn.microsoft.com/library/ee658247) and "Dynamic in C# 4.0: Introducing the ExpandoObject" on the C# FAQ blog (bit.ly/amRYRw).

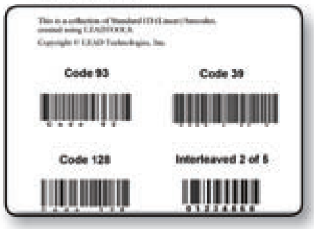
Class Wrappers

You can provide a better syntax for your own library or create a wrapper for an existing library. This is a more advanced scenario compared to the previous two and requires a deeper understanding of the DLR specifics.

Document

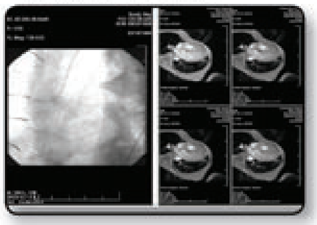


Barcode

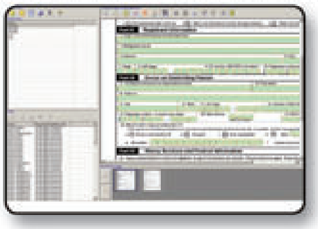


Multimedia

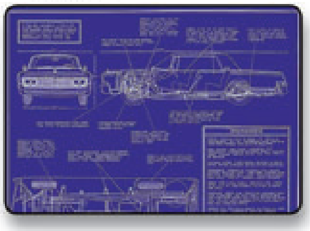
DICOM Medical



Form Recognition & Processing



Vector



For simple cases you can use the DynamicObject class. In this class you can mix static declaration of methods and properties with the dynamic dispatch. So you can store an object for which you want to provide better syntax in a class property, but handle all operations with this object through a dynamic dispatch.

As an example, look at the DynamicString class in **Figure 1** that wraps a string and displays names of all methods before actually calling those methods through reflection.

To instantiate this class you should use the dynamic keyword:

```
dynamic dStr = new DynamicString("Test");
Console.WriteLine(dStr.ToUpper());
Console.ReadLine();
```

Of course, this particular example is contrived and not really efficient. But if you have an API that already relies heavily on reflection, you can wrap all calls through reflection as shown here so the end users of your API won't see them.

> The DLR enables you to create scriptable applications because it provides a common set of hosting APIs for languages.

For more examples, refer to MSDN documentation (msdn.microsoft.com/library/system.dynamic.dynamicobject) and the "Dynamic in C# 4.0: Creating Wrappers with DynamicObject" post on the C# FAQ blog (bit.ly/dgS3od).

As I mentioned, the DynamicObject class provided by the DLR. DynamicObject or ExpandoObject is all you need to produce a dynamic object. However, some dynamic objects have a complicated binding logic for accessing members or invoking methods. Such objects need to implement the IDynamicMetaObjectProvider interface and provide their own dynamic dispatch. This is an advanced scenario and those who interested can read "Implementing Dynamic Interfaces" by Bill Wagner (msdn.microsoft.com/vcsharp/ff800651) and "Getting Started with the DLR as a Library Author" by Alex Turner and Bill Chiles (dlr.codeplex.com).

Scriptable Applications

Scripts are a powerful way to provide extensibility for your application. Microsoft Office can serve as a good example here: numerous macros, add-ons and plug-ins exist due to Visual Basic for Applications (VBA). And now the DLR enables you to create scriptable applications because it provides a common set of hosting APIs for languages.

For example, you can create an application where users would be able to add functionality themselves without asking for new features from the main product, like adding new characters and maps to a game or new graphs to a business application.

You have to use the open source version of the DLR from dlr.codeplex.com instead of the one used by the .NET Framework 4 because, right now, the DLR scripting and hosting APIs are only available in the open source version. Also, it's assumed that you'll write scripts not by using C#, but rather one of the .NET dynamic languages, such as IronPython or IronRuby. However, any language can support these APIs, even one that's not implemented on top of the DLR.

For details about using this functionality, watch the "Using Dynamic Languages to Build Scriptable Applications" presentation by Dino Viehland at PDC09 (microsoftpdc.com/2009/FT30).

Identifying Dynamic Objects

How can you distinguish dynamic objects from other objects? One easy way is to use built-in IDE features. You can either hover the mouse cursor over the object to see its declaration type or check whether IntelliSense is available (see **Figure 2**).

At run time, however, things get more complicated. You can't check whether the variable was declared by the dynamic keyword—the runtime type of the dynamic object is the type of the value it stores and you can't get its static type declaration. It's the same as if you declare your variable as object: At run time you can only get a type of the value that the variable holds; you can't tell whether this variable was originally declared as object.

What you can identify at run time is whether an object is coming from the DLR. It might be important because objects of types like ExpandoObject and DynamicObject can change their behavior at run time—for example, add and delete properties and methods.

Also, you can't use standard reflection methods to get information about such objects. If you add a property to an instance of the ExpandoObject class, you can't get this property from reflection:

```
dynamic expando = new ExpandoObject();
expando.SampleProperty = "This property was added at run time";
PropertyInfo dynamicProperty =
  expando.GetType().GetProperty("SampleProperty");
// dynamicProperty is null.
```

The good thing is that, in the .NET Framework 4, all objects that can dynamically add and remove members must implement one particular interface: System.Dynamic.IDynamicMetaObjectProvider. Both DynamicObject and ExpandoObject classes implement this interface, as well. However, this doesn't mean that any object declared by using the dynamic keyword implements this interface:

```
dynamic expando = new ExpandoObject();
Console.WriteLine(expando is IDynamicMetaObjectProvider);
// True

dynamic test = "test";
Console.WriteLine(test is IDynamicMetaObjectProvider);
// False
```

Figure 3 **Scripting Excel with C#**

```
// Add this line to the beginning of the file:
// using Excel = Microsoft.Office.Interop.Excel;

var excelApp = new Excel.Application();

excelApp.Workbooks.Add();
// excelApp.Workbooks.Add(Type.Missing);

excelApp.Visible = true;

Excel.Range targetRange = excelApp.Range["A1"];
// Excel.Range targetRange = excelApp.get_Range("A1", Type.Missing);

targetRange.Value = "Name";
// targetRange.set_Value(Type.Missing, "Name");

targetRange.Columns[1].AutoFit();
// ((Excel.Range)targetRange.Columns[1, Type.Missing]).AutoFit();
```

So, if you're using dynamic along with reflection, remember that reflection won't work for dynamically added properties and methods, and it might be a good idea to check whether the object you're reflecting on implements the IDynamicMetaObjectProvider interface.

Dynamic and COM Interop

The COM interop scenario that the C# team specifically targeted in the C# 4 release was programming against Microsoft Office applications, such as Word and Excel. The intent was to make this task as easy and natural in C# as it always was in Visual Basic. This is also a part of the Visual Basic and C# co-evolution strategy, where both languages aim at feature parity and borrow the best and most productive solutions from one another.

If you're interested in the details, read "C# and VB coevolution" on Scott Wiltamuth's Visual Studio blog (bit.ly/bFUpxG).

Figure 3 shows the C# 4 code that adds a value to the first cell in the Excel worksheet and then applies the AutoFit method to the first column. The comments under each line show equivalents from C# 3.0 and earlier.

The interesting thing about this example is that you can't see the dynamic keyword anywhere in the code. In fact, it's used in just one line here:

```
targetRange.Columns[1].AutoFit();
// ((Excel.Range)targetRange.Columns[1, Type.
Missing]).AutoFit();
```

In the C# 3.0 version, targetRange.Columns[1, Type.Missing] returns object, and that's why the casting to Excel.Range is necessary. But in C# 4 and Visual Studio 2010 such calls are silently converted into dynamic ones. So, the type of the targetRange.Columns[1] in C# 4 is actually dynamic.

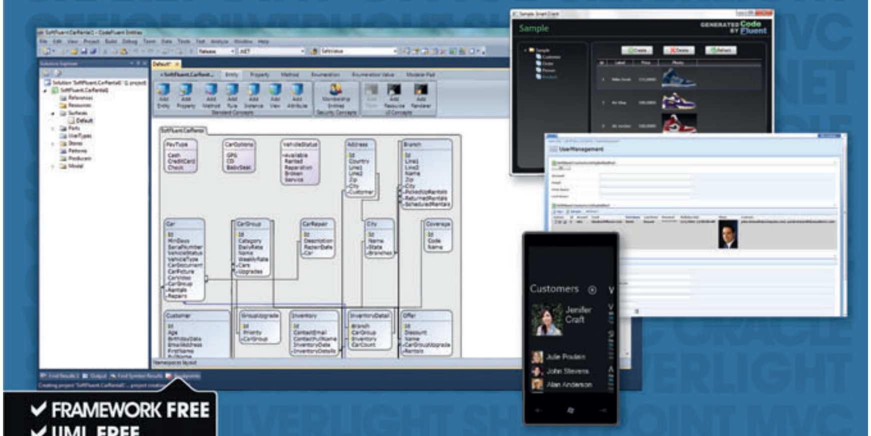
Another highlight is that the COM interop improvements in C# 4 are not just about dynamic. In all other lines better syntax is achieved because of other new features such as indexed properties and named and optional parameters. You can find a good overview of these new features in the *MSDN Magazine* article "New C# Features in the .NET Framework 4" by Chris Burrows (msdn.microsoft.com/magazine/ff796223).

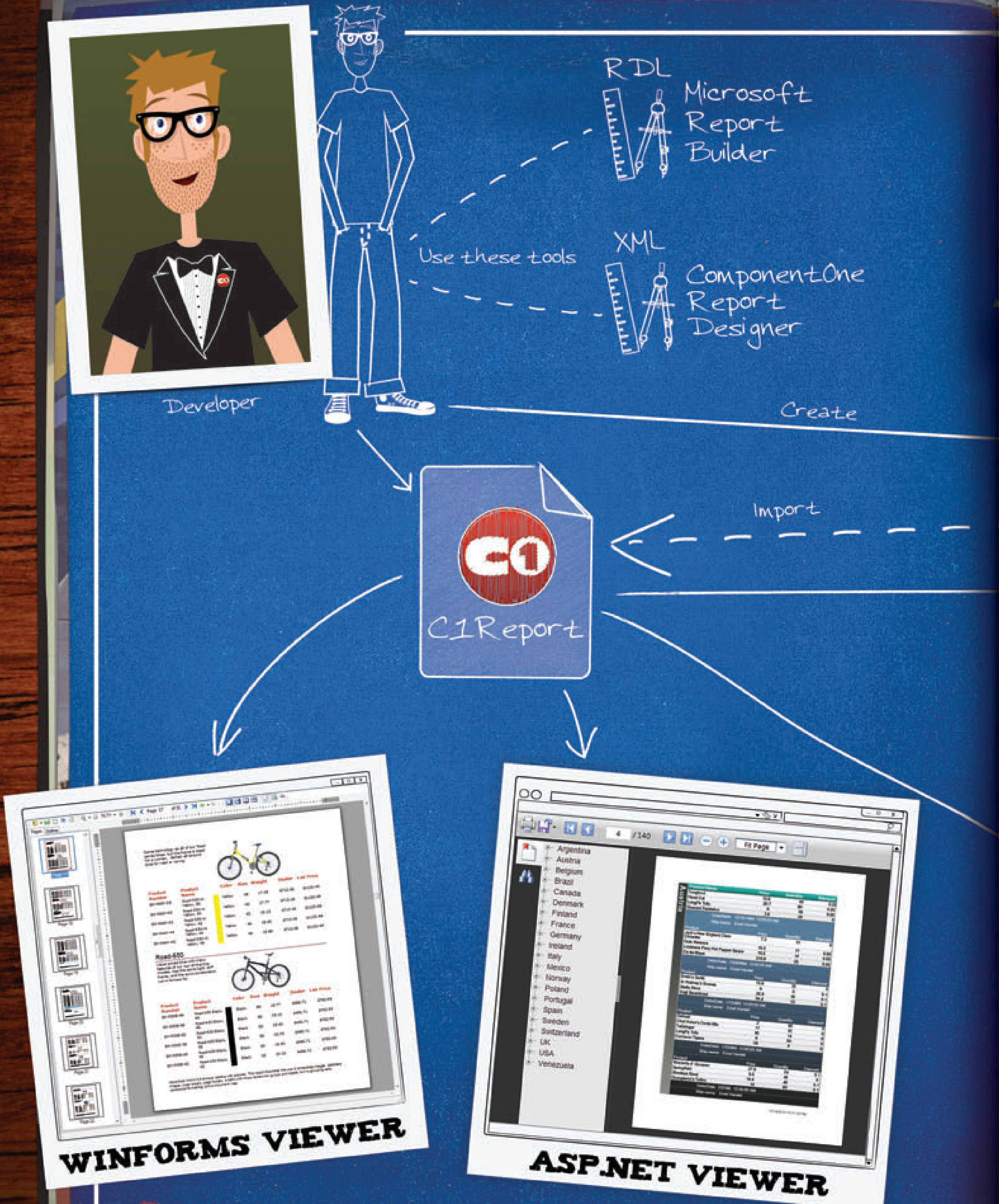
Where Can I Get More Information?

Hopefully this article covered most of the questions you might have about the dynamic keyword in C# 4, but I'm sure it didn't cover everything. If you have comments, questions or suggestions, drop by dlr.codeplex.com/discussions and ask away. Someone may have already asked about the issue, or you can create a new discussion. We have an active community and welcome new members. ■

ALEXANDRA RUSINA *is a program manager on the Silverlight team. Prior to that she worked as a programming writer on the Visual Studio Languages team during the Visual Studio 2010 release. She also regularly blogged on the C# Frequently Asked Question blog (blogs.msdn.com/b/csharpfaq/).*

THANKS to the following technical expert for reviewing this article: *Bill Chiles*

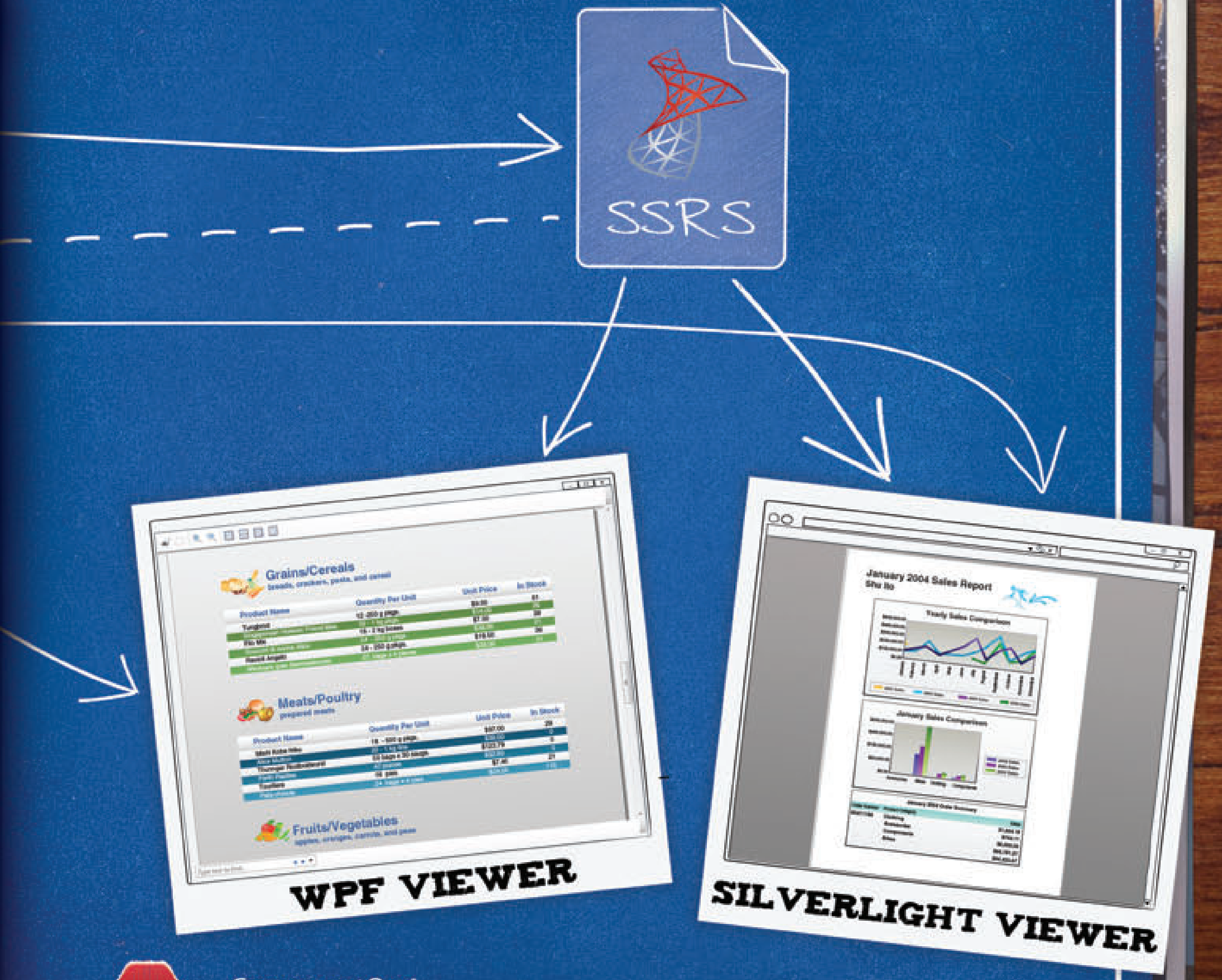




ComponentOne

YOUR GUIDE TO REPORTS

Developers, just like you, continue to turn to ComponentOne Reports™ for their enterprise solutions. Our flexible reports support Microsoft SQL Server Reporting Services (SSRS) with a fully customizable object model, report designers, schedulers, viewers for all .NET platforms, and more. Start your blueprint today; download ComponentOne Studio® Enterprise for high quality, professional apps.



ComponentOne®
Studio® Enterprise 2010

Download your free trial @
www.componentone.com/blueprint

Known Types and the Generic Resolver

Juval Lowy

Ever since its first release, Windows Communication Foundation (WCF) developers have had to deal with the hassles of data contract inheritance, a problem called known types. In this article I first explain the origin of the problem, discuss the available mitigations in the Microsoft .NET Framework 3.0 and the .NET Framework 4, and then present my technique that can eliminate the problem altogether. You'll also see some advanced WCF programming techniques.

By Value vs. by Reference

In traditional object-oriented languages such as C++ and C#, a derived class maintains an *is-a* relationship with its base class. This means that given this declaration, every B object is also an A object:

```
class A {...}
class B : A {...}
```

This article discusses:

- The known type crutches
- Data contract resolvers
- The generic resolver
- The generic resolver attribute

Technologies discussed:

Windows Communication Foundation

Code download available at:

code.msdn.microsoft.com/mag201102Resolver

Graphically, this looks like the Venn diagram in **Figure 1**, in which every B instance is also an A instance (but not every A is necessarily a B).

From a traditional object-orientation domain-modeling perspective, the is-a relationship enables you to design your code against the base class while interacting with a subclass. This means you can evolve the modeling of domain entities over time while minimizing the impact on the application.

For example, consider a business contacts management application with this modeling of a base type called Contact and a derived class called Customer that specializes the contact by adding to it the attributes of a customer:

```
class Contact {
  public string FirstName;
  public string LastName;
}

class Customer : Contact {
  public int OrderNumber;
}
```

Any method in the application that's written initially against the Contact type can accept Customer objects as well, as show in **Figure 2**.

The reason the code in **Figure 2** works at all has to do with the way the compiler represents the object state in memory. To support the is-a relationship between a subclass and its base class, when allocating a new subclass instance the compiler first allocates the base class portion of the state of the object, then appends directly after it the subclass portion, as shown in **Figure 3**.

When a method that expects a reference to a Contact is actually given a reference to a Customer, it still works because the Customer reference is a reference to a Contact as well.

Unfortunately, this intricate setup breaks when it comes to WCF. Unlike traditional object orientation or the classic CLR programming model, WCF passes all operation parameters by value, not by reference. Even though the code looks like the parameters are passed by reference (as in regular C#), the WCF proxy actually serializes the parameters into the message. The parameters are packaged in the WCF message and transferred to the service, where they are then deserialized to local references for the service operation to work with.

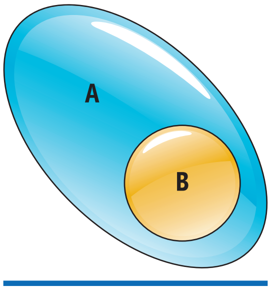


Figure 1 **Is-A Relationship**

This is also what happens when the service operation returns results to the client: The results (or outgoing parameters, or exceptions) are first serialized into a reply message and then deserialized back on the client side.

The exact form of the serialization that takes place is usually a product of the data contract the service contract is written against. For example, consider these data contracts:

```
[DataContract]
class Contact {...}

[DataContract]
class Customer : Contact {...}
```

Using these data contracts, you can define this service contract:

```
[ServiceContract]
interface IContactManager {
  [OperationContract]
  void AddContact(Contact contact);

  [OperationContract]
  Contact[] GetContacts();
}
```

With multitier applications, marshaling the parameters by value works better than by reference because any layer in the architecture is at liberty to provide its own interpretation to the behavior behind the data contract. Marshaling by value also enables remote calls, interoperability, queued calls and long-running workflows.

But unlike traditional object orientation, the service operation written against the Contact class can't by default work with the customer subclass. The reason is simple: If you do pass a subclass reference to a service operation that expects a base class reference, how would WCF know to serialize into the message the derived class portion?

As a result, given the definitions so far, this WCF code will fail:

```
class ContactManagerClient : ClientBase<IContactManager> :
  IContactManager{
  ...
}

IContactManager proxy = new ContactManagerClient();
Contact contact = new Customer();

// This will fail:
contacts.AddContact(contact);
```

The Known Type Crutches

With the .NET Framework 3.0, WCF was able to address the problem of substituting a base class reference with a subclass using the KnownTypeAttribute, defined as:

```
[AttributeUsage(AttributeTargets.Struct|AttributeTargets.Class,
  AllowMultiple = true)]
public sealed class KnownTypeAttribute : Attribute {
  public KnownTypeAttribute(Type type);
  //More members
}
```

The KnownType attribute allows you to designate acceptable subclasses for the data contract:

```
[DataContract]
[KnownType(typeof(Customer))]
class Contact {...}

[DataContract]
class Customer : Contact {...}
```

When the client passes a data contract that uses a known type declaration, the WCF message formatter tests the type (akin to using the is operator) and sees if it's the expected known type. If so, it serializes the parameter as the subclass rather than the base class.

The KnownType attribute affects all contracts and operations using the base class, across all services and endpoints, allowing it to accept subclasses instead of base classes. In addition, it includes the subclass in the metadata so that the client will have its own definition of the subclass and will be able to pass the subclass instead of the base class.

When multiple subclasses are expected, the developer must list all of them:

```
[DataContract]
[KnownType(typeof(Customer))]
[KnownType(typeof(Person))]
class Contact {...}

[DataContract]
class Person : Contact {...}
```

The WCF formatter uses reflection to collect all the known types of the data contracts, then examines the provided parameter to see if it's of any of the known types.

Note that you must explicitly add all levels in the data contract class hierarchy. Adding a subclass doesn't add its base classes:

```
[DataContract]
[KnownType(typeof(Customer))]
[KnownType(typeof(Person))]
class Contact {...}

[DataContract]
class Customer : Contact {...}

[DataContract]
class Person : Customer {...}
```

Because the KnownType attribute may be too broad in scope, WCF also provides ServiceKnownTypeAttribute, which you can apply on a specific operation or on a specific contract.

Figure 2 **Interchanging Base Class and Sub Class References**

```
interface IContactManager {
  void AddContact(Contact contact);
  Contact[] GetContacts();
}

class AddressBook : IContactManager {
  public void AddContact(Contact contact)
  {...}
  ...
}

IContactManager contacts = new AddressBook();

Contact  contact1 = new Contact();
Contact  contact2 = new Customer();
Customer customer = new Customer();

contacts.AddContact(contact1);
contacts.AddContact(contact2);
contacts.AddContact(customer);
```

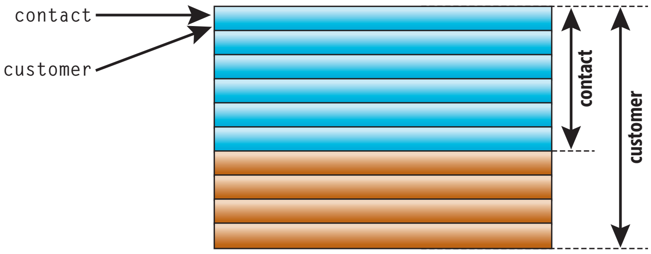


Figure 3 **Object State Hierarchy in Memory**

Finally, in the .NET Framework 3.0, WCF also allowed listing the expected known types in the application config file in the system.runtime.serialization section.

While using known types technically works just fine, you should feel some unease about it. In traditional object-oriented modeling you never want to couple the base class to any specific subclasses. The hallmark of a good base class is precisely that: a good base is a good base class for any possible subclass, and yet the known types issue makes it adequate only for subclasses it happens to know about. If you do all your modeling up-front when designing the system, that may not be a hindrance. In reality, over time, as the application evolves its modeling, you'll encounter as-yet-unknown types that will force you to, at the very least, redeploy your application—and, more likely, to also modify your base classes.

Data Contract Resolvers

To alleviate the problem, in the .NET Framework 4 WCF introduced a way of resolving the known types at run time. This programmatic technique, called data contract resolvers, is the most powerful option because you can extend it to completely automate dealing with the known type issues. In essence, you're given a chance to intercept the operation's attempt to serialize and deserialize parameters and resolve the known types at run time both on the client and service sides.

The first step in implementing a programmatic resolution is to derive from the abstract class DataContractResolver, defined as:

```
public abstract class DataContractResolver {
  protected DataContractResolver();

  public abstract bool TryResolveType(
    Type type,Type declaredType,
    DataContractResolver knownTypeResolver,
    out XmlDictionaryString typeName,
    out XmlDictionaryString typeNamespace);

  public abstract Type ResolveName(
    string typeName,string typeNamespace,
    Type declaredType,
    DataContractResolver knownTypeResolver);
}
```

Your implementation of TryResolveType is called when WCF tries to serialize a type into a message and the type provided (the type parameter) is different from the type declared in the operation contract (the declaredType parameter). If you want to serialize the type, you need to provide some unique identifiers to serve as keys into a dictionary that maps identifiers to types. WCF will provide those keys during deserialization so that you can bind against that type.

Note that the namespace key can't be an empty string or a null. While virtually any unique string value will do for the identifiers, I recommend simply using the CLR type name and namespace. Set the type name and namespace into the typeName and typeNamespace out parameters.

If you return true from TryResolveType, the type is considered resolved, as if you had applied the KnownType attribute. If you return false, WCF fails the call. Note that TryResolveType must resolve all known types, even those types that are decorated with the KnownType attribute or are listed in the config file. This presents a potential risk: It requires the resolver to be coupled to all known types in the application and will fail the operation call with other types that may come over time. It's therefore preferable as a fall-back contingency to try to resolve the type using the default known types resolver that WCF would've used if your resolver was not in use. This is exactly what the knownTypeResolver parameter is for. If your implementation of TryResolveType can't resolve the type, it should delegate to knownTypeResolver.

ResolveName is called when WCF tries to deserialize a type out of a message and the type provided (the type parameter) is different from the type declared in the operation contract (the declaredType parameter). In this case, WCF provides the type name and namespace identifiers so that you can map them back to a known type.

Figure 4 **The CustomerResolver**

```
class CustomerResolver : DataContractResolver {
  string Namespace {
    get {
      return typeof(Customer).Namespace ?? "global";
    }
  }

  string Name {
    get {
      return typeof(Customer).Name;
    }
  }

  public override Type ResolveName(
    string typeName,string typeNamespace,
    Type declaredType,
    DataContractResolver knownTypeResolver) {

    if(typeName == Name && typeNamespace == Namespace) {
      return typeof(Customer);
    }
    else {
      return knownTypeResolver.ResolveName(
        typeName,typeNamespace,declaredType,null);
    }
  }

  public override bool TryResolveType(
    Type type,Type declaredType,
    DataContractResolver knownTypeResolver,
    out XmlDictionaryString typeName,
    out XmlDictionaryString typeNamespace) {

    if(type == typeof(Customer)) {
      XmlDictionary dictionary = new XmlDictionary();
      typeName      = dictionary.Add(Name);
      typeNamespace = dictionary.Add(Namespace);
      return true;
    }
    else {
      return knownTypeResolver.TryResolveType(
        type,declaredType,null,out typeName,out typeNamespace);
    }
  }
}
```

Powerful Tools for Developers

v4.5!

High-Performance PDF Printer Driver

- Create accurate PDF documents in a fraction of the time needed with other tools
- WHQL tested for all Windows 32 and 64-bit platforms
- Produce fully compliant PDF/A documents
- Standard PDF features included with a number of unique features
- Interface with any .NET or ActiveX programming language

v4.5!

PDF Editor for .NET, now Webform Enabled

- Edit, process and print PDF 1.7 documents programmatically
- Fast and lightweight 32 and 64-bit managed code assemblies for Windows, WPF and Web applications
- Support for dynamic objects such as edit-fields and sticky-notes
- Save image files directly to PDF, with optional OCR
- Multiple image compression formats such as PNG, JBIG2 and TIFF

New!

PDF Integration into Silverlight Applications

- Server-side PDF component based on the robust Amyuni PDF Creator ActiveX or .NET components
- Client-side C# Silverlight 3 control provided with source-code
- Optimization of PDF documents prior to converting them into XAML
- Conversion of PDF edit-boxes into Silverlight TextBox objects
- Support for other document formats such as TIFF and XPS

New Touchscreen Tablet for Mobile Development!

The DevTouch Pro is a new color touchscreen tablet designed to provide mobile application developers with a customizable development, testing and deployment platform.

Microsoft Windows CE android Linux

- Fully open customizable tablet
- Develop with .NET, Java or C++
- Unrestricted development and flexible quantities
- Fully supported in North America

WiFi Bluetooth MADE IN CANADA FABRIQUÉ AU CANADA

Learn more at www.devtouchpro.com

Windows 7 CERTIFIED FOR Windows Server 2008

More Development Tools Available at:

www.amyuni.com

USA and Canada
Toll Free: 1 866 926 9864
Support: (514) 868 9227

Info: sales@amyuni.com

Europe
Sales: (+33) 1 30 61 07 97
Support: (+33) 1 30 61 07 98

Customizations: management@amyuni.com

AMYUNI Technologies

As an example, consider again these two data contracts:

```
[DataContract]
class Contact {...}

[DataContract]
class Customer : Contact {...}
```

Figure 4 lists a simple resolver for the Customer type.

The resolver must be attached as a behavior for each operation on the proxy or the service endpoint. The ServiceEndpoint class has a property called Contract of the type ContractDescription:

```
public class ServiceEndpoint {
  public ContractDescription Contract
  {get;set;}

  // More members
}
```

ContractDescription has a collection of operation descriptions, with an instance of OperationDescription for every operation on the contract:

```
public class ContractDescription {
  public OperationDescriptionCollection Operations
  {get;}

  // More members
}
public class OperationDescriptionCollection :
  Collection<OperationDescription>
{...}
```

Each OperationDescription has a collection of operation behaviors of the type IOperationBehavior:

```
public class OperationDescription {
  public KeyedByTypeCollection<IOperationBehavior> Behaviors
  {get;}
  // More members
}
```

In its collection of behaviors, every operation always has a behavior called DataContractSerializerOperationBehavior with a DataContractResolver property:

```
public class DataContractSerializerOperationBehavior :
  IOperationBehavior,... {
  public DataContractResolver DataContractResolver
  {get;set}
  // More members
}
```

The DataContractResolver property defaults to null, but you can set it to your custom resolver. To install a resolver on the host side, you must iterate over the collection of endpoints in the service description maintained by the host:

```
public class ServiceHost : ServiceHostBase {...}

public abstract class ServiceHostBase : ... {
  public ServiceDescription Description
  {get;}
  // More members
}

public class ServiceDescription {
  public ServiceEndpointCollection Endpoints
  {get;}
  // More members
}

public class ServiceEndpointCollection :
  Collection<ServiceEndpoint> {...}
```

Suppose you have the following service definition and are using the resolver in **Figure 4**:

```
[ServiceContract]
interface IContactManager {
  [OperationContract]
  void AddContact(Contact contact);
  ...
}
class AddressBookService : IContactManager {...}
```

Figure 5 **Installing a Resolver on the Host**

```
ServiceHost host =
  new ServiceHost(typeof(AddressBookService));

foreach(ServiceEndpoint endpoint in
  host.Description.Endpoints) {
  foreach(OperationDescription operation in
    endpoint.Contract.Operations) {

    DataContractSerializerOperationBehavior behavior =
      operation.Behaviors.Find<
        DataContractSerializerOperationBehavior>();
      behavior.DataContractResolver = new CustomerResolver();
  }
}
host.Open();
```

Figure 5 shows how to install the resolver on the host for the AddressBookService.

On the client side, you follow similar steps, except you need to set the resolver on the single endpoint of the proxy or the channel factory. For example, given this proxy class definition:

```
class ContactManagerClient : ClientBase<IContactManager>,IContactManager
{...}
```

Figure 6 shows how to install the resolver on the proxy in order to call the service of **Figure 5** with a known type.

The Generic Resolver

Writing and installing a resolver for each type is obviously a lot of work, requiring you to meticulously track all known types—something that's error-prone and can quickly get out of hand in an evolving system. To automate implementing a resolver, I wrote the class GenericResolver, defined as:

```
public class GenericResolver : DataContractResolver {
  public Type[] KnownTypes
  {get;}

  public GenericResolver();
  public GenericResolver(Type[] typesToResolve);

  public static GenericResolver Merge(
    GenericResolver resolver1,
    GenericResolver resolver2);
}
```

GenericResolver offers two constructors. One constructor can accept an array of known types to resolve. The parameterless constructor will automatically add as known types all classes and structs in the calling assembly and all public classes and structs in assemblies referenced by the calling assembly. The parameterless

Figure 6 **Installing a Resolver on the Proxy**

```
ContactManagerClient proxy = new ContactManagerClient();

foreach(OperationDescription operation in
  proxy.Endpoint.Contract.Operations) {

  DataContractSerializerOperationBehavior behavior =
    operation.Behaviors.Find<
    DataContractSerializerOperationBehavior>();

  behavior.DataContractResolver = new CustomerResolver();
}

Customer customer = new Customer();
...

proxy.AddContact(customer);
```

MORE THAN JUST
TRADITIONAL REPORTING

ASPOSE...
Your File Format Experts

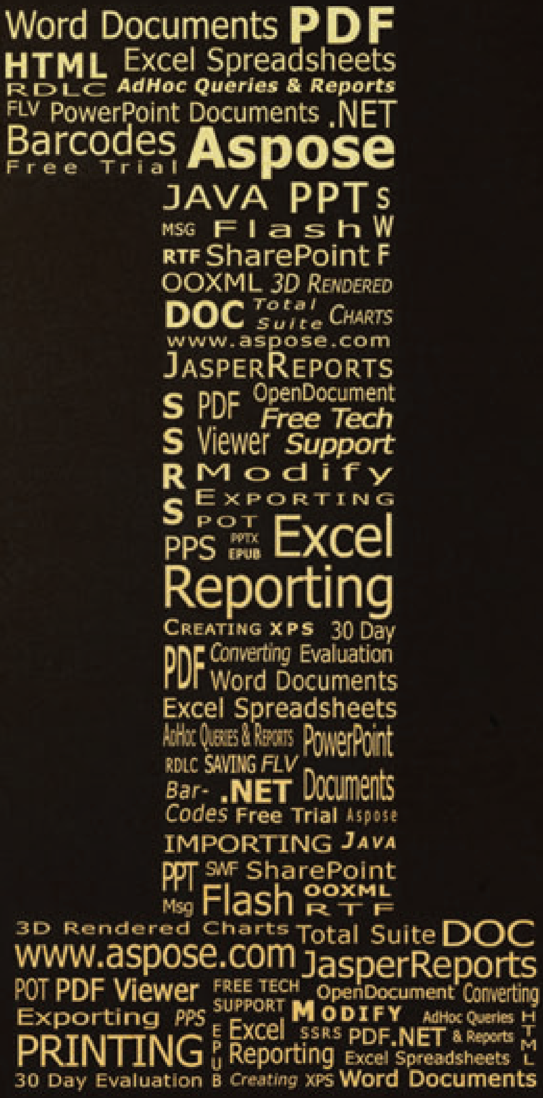
CONVERT, IMPORT, EXPORT PRINT & MODIFY

Files Like:

DOCX PDF PPT ODF
XLSX SWF MPP MSG
Barcode Report InfoPath

File management solutions for:

- ➤ .NET
- ➤ Java
- ➤ SSRS Rendering Extensions
- ➤ SharePoint
- ➤ JasperReports Exporters


Word Documents PDF
HTML Excel Spreadsheets
RDLC AdHoc Queries & Reports
FLV PowerPoint Documents .NET
Barcodes Aspose
Free Trial
JAVA PPTS
MSG Flash W
RTF SharePoint F
OOXML 3D Rendered
DOC Total Suite CHARTS
www.aspose.com
JasperReports
S PDF OpenDocument Free Tech
S Viewer Support
R Modify
S Exporting
POT
PPS PPTX EPUB Excel
Reporting
Creating XPS 30 Day
PDF Converting Evaluation
Word Documents
Excel Spreadsheets
AdHoc Queries & Reports PowerPoint
RDLC SAVING FLV
Bar- .NET Documents
Codes Free Trial Aspose
IMPORTING Java
PPT SWF SharePoint
Msg Flash OOXML
RTF
3D Rendered Charts Total Suite DOC
www.aspose.com JasperReports
POT PDF Viewer FREE TECH SUPPORT OpenDocument Converting
Exporting PPS MODIFY AdHoc Queries & Reports
EPUB Excel SSRS PDF.NET HTML
PRINTING Reporting Excel Spreadsheets
30 Day Evaluation Creating XPS Word Documents

Try Aspose FREE for 30 days!

Get your FREE evaluation copy at http://www.aspose.com

ASPOSE
Your File Format Experts

US Sales: 1.888.277.6734 · EU Sales: +44 (0)800 098 8425 · email: sales@aspose.com · enterprise.sales@aspose.com

Figure 7 **Implementing GenericResolver (Partial)**

```
public class GenericResolver : DataContractResolver {
  const string DefaultNamespace = "global";

  readonly Dictionary<Type,Tuple<string,string>> m_TypeToNames;
  readonly Dictionary<string,Dictionary<string,Type>> m_NamesToType;

  public Type[] KnownTypes {
    get {
      return m_TypeToNames.Keys.ToArray();
    }
  }

  // Get all types in calling assembly and referenced assemblies
  static Type[] ReflectTypes() {...}

  public GenericResolver() : this(ReflectTypes()) {}

  public GenericResolver(Type[] typesToResolve) {
    m_TypeToNames = new Dictionary<Type,Tuple<string,string>>();
    m_NamesToType = new Dictionary<string,Dictionary<string,Type>>();

    foreach(Type type in typesToResolve) {
      string typeNamespace = GetNamespace(type);
      string typeName = GetName(type);

      m_TypeToNames[type] = new Tuple<string,string>(typeNamespace,typeName);

      if(m_NamesToType.ContainsKey(typeNamespace) == false) {
        m_NamesToType[typeNamespace] = new Dictionary<string,Type>();
      }

      m_NamesToType[typeNamespace][typeName] = type;
    }
  }

  static string GetNamespace(Type type) {
    return type.Namespace ?? DefaultNamespace;
  }

  static string GetName(Type type) {
    return type.Name;
  }

  public static GenericResolver Merge(
    GenericResolver resolver1, GenericResolver resolver2) {

    if(resolver1 == null) {
      return resolver2;
    }

    if(resolver2 == null) {
      return resolver1;
    }

    List<Type> types = new List<Type>();

    types.AddRange(resolver1.KnownTypes);
    types.AddRange(resolver2.KnownTypes);

    return new GenericResolver(types.ToArray());
  }

  public override Type ResolveName(
    string typeName,string typeNamespace,
    Type declaredType,
    DataContractResolver knownTypeResolver) {

    if(m_NamesToType.ContainsKey(typeNamespace)) {
      if(m_NamesToType[typeNamespace].ContainsKey(typeName)) {
        return m_NamesToType[typeNamespace][typeName];
      }
    }

    return knownTypeResolver.ResolveName(
      typeName,typeNamespace,declaredType,null);
  }

  public override bool TryResolveType(
    Type type,Type declaredType,
    DataContractResolver knownTypeResolver,
    out XmlDictionaryString typeName,
    out XmlDictionaryString typeNamespace) {

    if(m_TypeToNames.ContainsKey(type)) {
      XmlDictionary dictionary = new XmlDictionary();
      typeNamespace = dictionary.Add(m_TypeToNames[type].Item1);
      typeName      = dictionary.Add(m_TypeToNames[type].Item2);
      return true;
    }
    else {
      return knownTypeResolver.TryResolveType(
      type,declaredType,null,out typeName,
      out typeNamespace);
    }
  }
}
```

constructor won't add types originating in a .NET Framework-referenced assembly.

In addition, GenericResolver offers the Merge static method that you can use to merge the known types of two resolvers, returning a GenericResolver that resolves the union of the two resolvers provided. **Figure 7** shows the pertinent portion of GenericResolver without reflecting the types in the assemblies, which has nothing to do with WCF.

The most important members of GenericResolver are the m_TypeToNames and the m_NamesToType dictionaries. m_TypeToNames maps a type to a tuple of its name and namespace. m_NamesToType maps a type namespace and name to the actual type. The constructor that takes the array of types initializes those two dictionaries. The TryResolveType method uses the provided type as a key into the m_TypeToNames dictionary to read the type's name and namespace. The ResolveName method uses the provided namespace and name as keys into the m_NamesToType dictionary to return the resolved type.

While you could use tedious code similar to **Figure 5** and **Figure 6** to install GenericResolver, it's best to streamline it with extension methods. To that end, use my AddGenericResolver methods of GenericResolverInstaller, defined as:

```
public static class GenericResolverInstaller {
  public static void AddGenericResolver(
    this ServiceHost host, params Type[] typesToResolve);

  public static void AddGenericResolver<T>(
    this ClientBase<T> proxy,
    params Type[] typesToResolve) where T : class;

  public static void AddGenericResolver<T>(
    this ChannelFactory<T> factory,
    params Type[] typesToResolve) where T : class;
}
```

The AddGenericResolver method accepts a params array of types, which means an open-ended, comma-separated list of types. If you don't specify types, that will make AddGenericResolver add as known types all classes and structs in the calling assembly plus the public classes and structs in referenced assemblies. For example, consider these known types:

```
[DataContract]
class Contact {...}

[DataContract]
class Customer : Contact {...}

[DataContract]
class Employee : Contact {...}
```

Figure 8 **Installing GenericResolver**

```
// Host side

ServiceHost host1 = new ServiceHost(typeof(AddressBookService));
// Resolve all types in this and referenced assemblies
host1.AddGenericResolver();
host1.Open();

ServiceHost host2 = new ServiceHost(typeof(AddressBookService));
// Resolve only Customer and Employee
host2.AddGenericResolver(typeof(Customer),typeof(Employee));
host2.Open();

ServiceHost host3 = new ServiceHost(typeof(AddressBookService));
// Can call AddGenericResolver() multiple times
host3.AddGenericResolver(typeof(Customer));
host3.AddGenericResolver(typeof(Employee));
host3.Open();

// Client side

ContactManagerClient proxy = new ContactManagerClient();
// Resolve all types in this and referenced assemblies
proxy.AddGenericResolver();

Customer customer = new Customer();
...
proxy.AddContact(customer);
```

Figure 8 shows several examples of using the AddGenericResolver extension method for these types.

GenericResolverInstaller not only installs the GenericResolver, it also tries to merge it with the old generic resolver (if present). This means you can call the AddGenericResolver method multiple times. This is handy when adding bounded generic types:

```
[DataContract]
class Customer<T> : Contact {...}

ServiceHost host = new ServiceHost(typeof(AddressBookService));

// Add all non-generic known types
host.AddGenericResolver();

// Add the generic types
host.AddGenericResolver(typeof(Customer<int>,Customer<string>));

host.Open();
```

Figure 9 shows partial implementation of GenericResolverInstaller.

If no types are provided, AddGenericResolver will use the parameterless constructor of GenericResolver. Otherwise, it will use only the specified types by calling the other constructor. Note the merging with the old resolver if present.

The Generic Resolver Attribute

If your service relies on the generic resolver by design, it's better not to be at the mercy of the host and to declare your need for the generic resolver at design time. To that end, I wrote the GenericResolverBehaviorAttribute:

```
[AttributeUsage(AttributeTargets.Class)]
public class GenericResolverBehaviorAttribute :
  Attribute,IServiceBehavior {

  void IServiceBehavior.Validate(
    ServiceDescription serviceDescription,
    ServiceHostBase serviceHostBase) {

    ServiceHost host = serviceHostBase as ServiceHost;
    host.AddGenericResolver();
  }
  // More members
}
```

Figure 9 **Implementing GenericResolverInstaller**

```
public static class GenericResolverInstaller {
  public static void AddGenericResolver(
    this ServiceHost host, params Type[] typesToResolve) {

    foreach(ServiceEndpoint endpoint in
      host.Description.Endpoints) {

      AddGenericResolver(endpoint,typesToResolve);
    }
  }

  static void AddGenericResolver(
    ServiceEndpoint endpoint,Type[] typesToResolve) {

    foreach(OperationDescription operation in
      endpoint.Contract.Operations) {

      DataContractSerializerOperationBehavior behavior =
        operation.Behaviors.Find<
        DataContractSerializerOperationBehavior>();

      GenericResolver newResolver;

      if(typesToResolve == null ||
        typesToResolve.Any() == false) {

        newResolver = new GenericResolver();
      }
      else {
        newResolver = new GenericResolver(typesToResolve);
      }

      GenericResolver oldResolver =
        behavior.DataContractResolver as GenericResolver;
      behavior.DataContractResolver =
        GenericResolver.Merge(oldResolver,newResolver);
    }
  }
}
```

This concise attribute makes the service independent of the host:

```
[GenericResolverBehavior]
class AddressBookService : IContactManager {...}
```

GenericResolverBehaviorAttribute derives from IServiceBehavior, which is a special WCF interface and is the most commonly used extension in WCF. When the host loads the service, the host calls the IServiceBehavior methods—specifically the Validate method—which lets the attribute interact with the host. In the case of GenericResolverBehaviorAttribute, it adds the generic resolver to the host.

GenericResolver offers two constructors.

And there you have it: a relatively simple and flexible way to bypass the hassles of data contract inheritance. Put this technique to work in your next WCF project. ■

JUVAL LOWY *is a software architect with IDesign providing .NET and architecture training and consulting. This article contains excerpts from his recent book, "Programming WCF Services, 3rd Edition" (O'Reilly, 2010). He's also the Microsoft Regional Director for the Silicon Valley. Contact Lowy at idesign.net.*

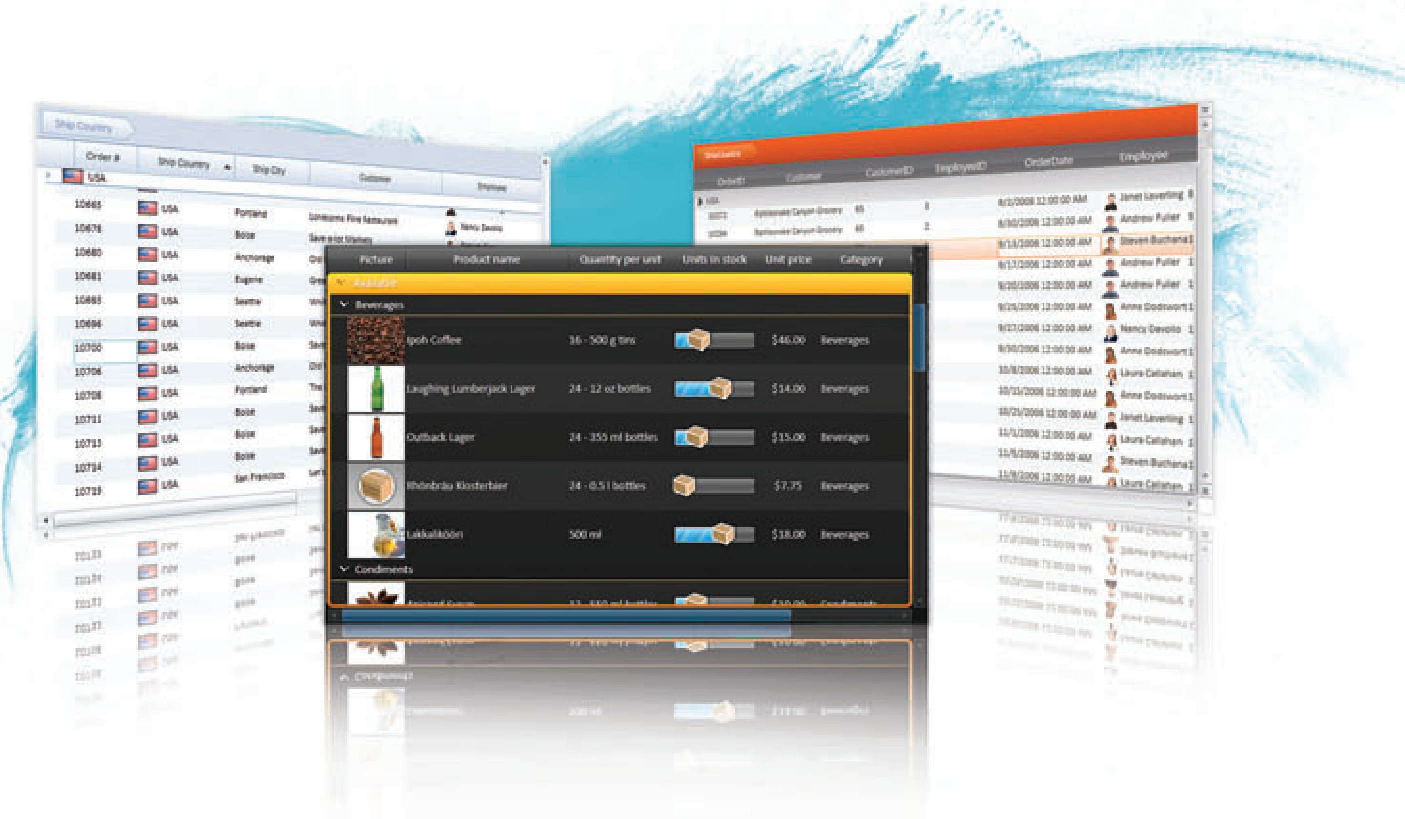
THANKS to the following technical experts for reviewing this article: *Glenn Block and Amadeo Casas Cuadrado*

Datagrid speed, Xceed style

XCEED
DataGrid
for Silverlight

The only Silverlight datagrid that:

- Lets end-users access remote data as fast as local.
- Uses an intelligent background data retrieval system.
- Saves end-users from experiencing lag in the UI.



Ship Country

Order #	Ship Country	Ship City	Customer	Employee
USA				
10665	USA	Portland	Lonesome Pine Restaurant	
10678	USA	Boise	Save-a-lot Markets	Nancy Davolio
10680	USA	Anchorage		
10681	USA	Eugene		
10685	USA	Seattle		
10696	USA	Seattle		
10700	USA	Boise		
10706	USA	Anchorage		
10708	USA	Portland		
10711	USA	Boise		
10713	USA	Boise		
10714	USA	Boise		
10719	USA	San Francisco		

Picture	Product name	Quantity per unit	Units in stock	Unit price	Category
Available					
Beverages					
	Ipoh Coffee	16 - 500 g tins		$46.00	Beverages
	Laughing Lumberjack Lager	24 - 12 oz bottles		$14.00	Beverages
	Outback Lager	24 - 355 ml bottles		$15.00	Beverages
	Rhönbräu Klosterbier	24 - 0.5 l bottles		$7.75	Beverages
	Lakkalikööri	500 ml		$18.00	Beverages
Condiments					

ShipCountry

OrderID	Customer	CustomerID	EmployeeID	OrderDate	Employee
USA					
[illegible]	Rattlesnake Canyon Grocery	65	3	8/1/2006 12:00:00 AM	Janet Leverling
[illegible]	Rattlesnake Canyon Grocery	65	2	8/30/2006 12:00:00 AM	Andrew Fuller
				9/13/2006 12:00:00 AM	Steven Buchana
				9/17/2006 12:00:00 AM	Andrew Fuller
				9/20/2006 12:00:00 AM	Andrew Fuller
				9/25/2006 12:00:00 AM	Anna Dodswort
				9/27/2006 12:00:00 AM	Nancy Davolio
				9/30/2006 12:00:00 AM	Anna Dodswort
				10/8/2006 12:00:00 AM	Laura Callahan
				10/15/2006 12:00:00 AM	Anna Dodswort
				10/25/2006 12:00:00 AM	Janet Leverling
				11/1/2006 12:00:00 AM	Laura Callahan
				11/5/2006 12:00:00 AM	Steven Buchana
				11/8/2006 12:00:00 AM	Laura Callahan

There's fast and then there's Xceed fast. Going beyond current-generation Silverlight datagrids, Xceed DataGrid for Silverlight is fast where others are slow: when working with remote data. With advanced, background record retrieval from remote sources and a responsive UI that minimizes perceived lag, the control keeps the end-user experience smooth and snappy, no matter where the data is.

Microsoft® Visual Studio® PARTNER

Datagrids, transformed

Try it live on xceed.com

XCEED
MULTI-TALENTED COMPONENTS

Pattern Matching Database Records with F#

Ambar Ray

The data used by your applications doesn't just appear out of thin air. To end up with a database full of useful data, you're going to have to jump through a few hoops. To start, you probably perform an extract, transform and load (ETL) process on some collection of dimensional data. This typically includes cleansing, pruning and standardizing the data. This is just to get the data into a form that works in your database.

After the ETL step, you'll want to go through your records and make sure they're useful and consistent. This typically means implementing a matching and deduplication process. Next, you'll do some name and address parsing. With this information you can begin a matching process and start identifying duplicates.

There are four common matching algorithms used for attribute deduplication processes: absolute match, partial match, Soundex and lookup match. These algorithms can be run against the data and, once the percentage match score is computed, you can decide whether to discard or store the data.

This article discusses:

- Preparing data for matching
- Matching and deduplication basics
- The four matching patterns
- Scoring the results

Technologies discussed:

F#, SQL Server

Code download available at:

code.msdn.microsoft.com/mag201102FSharp

As an exercise, I've implemented these four matching algorithms using F# pattern matching and asynchronous programming features to quickly calculate the aggregate match score. In this article, I'll show you my implementation of the matching algorithms and how I create the aggregate match score. The ultimate objective of this article is to showcase some of the features of F# such as functional programming, imperative programming and implicit type inference system, and demonstrate how you can use F# for accomplishing some significant data management tasks quickly and easily.

Preparing the Data

I'll start by loading the dimensional data from various transactional systems in a staging table. The data source could be a relational database, a flat file, an Excel file or an XML file. The ETL process often uses a tool like SQL Server Integration Services (SSIS) 2008 for cleansing, pruning and standardizing the incoming data and subsequently loading the staging tables. The staging tables and the master database are kept in a master data hub.

As mentioned earlier, I use a separate application built with F# and Windows Presentation Foundation (WPF) to take care of the matching and deduplication process. In actual projects, you'd first generate ADO.NET Entity Framework models in the data-access layer. These models model the master data tables along with all the lookup tables present in the master data hub.

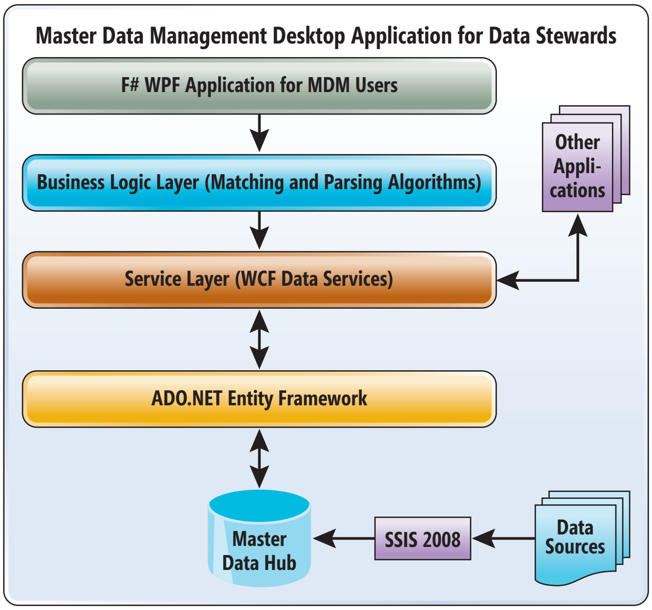

Master Data Management Desktop Application for Data Stewards
F# WPF Application for MDM Users
Business Logic Layer (Matching and Parsing Algorithms)
Other Appli-cations
Service Layer (WCF Data Services)
ADO.NET Entity Framework
Master Data Hub
SSIS 2008
Data Sources

Figure 1 **Master Data Management Application Architecture**

Next, the layer above should expose the underlying model through Windows Communication Foundation (WCF) services. The business logic layer above would implement the matching and deduplication routines as well as the parsing methods. Finally, the presentation layer would present various GUIs to the data manager. **Figure 1** depicts the architecture of the application.

I'm not going to demonstrate the entire application here, only the implementation of the matching and parsing algorithms. The subsequent sections will explain the implementation of the matching and parsing algorithms in the business logic layer.

Getting Started

For the purposes of this scenario, let's assume that the row data are retrieved from the database via the data-access layer and WCF Data Services is subsequently stored in List<T> objects for further processing. These List<T> objects will be populated with test data for implementing the pertinent algorithms.

The master database contains all the source tables as well as staging and history tables. As an example, **Figure 2** shows the composition of a Customer data entity.

The matching algorithm can be explained by means of the flowchart in **Figure 3**.

Step 1 is not really part of the matching process, but rather a precursor. Data is cleansed and standardized to a preferred format before it's sent for parsing.

Name Parsing

The parsing process in Step 2 is also a precursor to the matching process. In this step, individual names are parsed out of the data. Assuming that a salutation may be entered along with the first name, it should be broken down (parsed) into individual elements: salutation and first name. Thus, if Mr. John is provided in the first name field, with Mr. being the salutation and John being the actual first name, then the algorithm will work as follows:

1. Pick the first word terminated with a space (position A) as W1.
2. Identify whether W1 is a salutation by matching with a lookup list. If so, ignore the salutation and proceed to step 4.
3. If W1 is not a salutation, consider it the first name. Do not consider the string snippet for further parsing.
4. If W1 is identified as a Salutation, identify the next occurrence of a space or an end of string (Position B). Consider the word enclosed within position A and position B as First Name. Do not consider the string snippet for further parsing.

For example, the string "Mr. John" would be parsed as shown in **Figure 4**.

Position 1 to position 3 would constitute the salutation. Salutation is optional, and would be populated only if the first word matches with an exhaustive list of possible values. (I do make an assumption here that each part of the name, including salutation, has to be followed by a space. An exception is made for the last name field.) In **Figure 4**, position 5 to position 8 would constitute the first name.

The implementation of this parsing algorithm in F# looks like this:

```
let ParseName(strName : string)=
  let input = strName.Trim()
  let splitNames = input.Split([|" "|], StringSplitOptions.None)
  match splitNames.[0] with
    | "Mr" | "Mr." | "Mrs" | "Mrs." | "Dr" | "Dr." | "Kum" | "Kum."
      -> splitNames.[1]
    | _-> splitNames.[0]
```

In this code I've done pattern matching using the "match" and "with" keywords of F#, followed by pattern matching rules, each followed by the "->" symbol.

Address Parsing

The next step is address line parsing. Address lines 1 and 2 (concatenated) should be broken down (parsed) into a house or building number, street name, street type, apartment type (optional, constituting Apt, Flat or Suite) and apartment number (optional). If city, state and country information are provided in address lines 1 and 2, then those need to be omitted.

A combined address should be parsed into a house number, street name, street type and, if applicable, an apartment type and apartment number. To illustrate, let's look at an example:

421, East Drachman St, Suite 5A, Tucson, Arizona, beside McDonald's

In this case the address elements are parsed as shown in **Figure 5**.

Figure 2 **Customer Entity**

CUSTFIRSTNAME
CUSTLASTNAME
CUSTHOUSENO
CUSTSTREETNAME
CUSTSTREETTYPE
CUSTCITY
CUSTSTATE
CUSTPOSTCODE
CUSTMOBILE
CUSTCOUNTRY
CUSTID
CUSTACTIVEFLAG

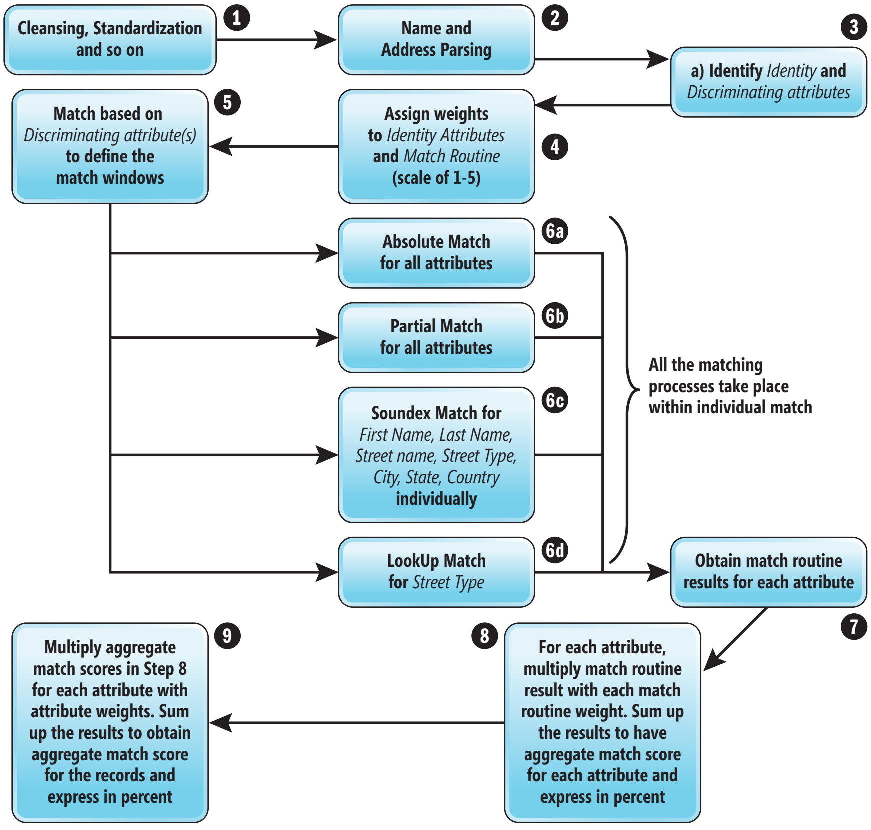

Cleansing, Standardization and so on
1
Name and Address Parsing
2
3
a) Identify *Identity* and *Discriminating attributes*
Match based on *Discriminating attribute(s)* to define the match windows
5
Assign weights to *Identity Attributes* and *Match Routine* (scale of 1-5)
4
Absolute Match for all attributes
6a
Partial Match for all attributes
6b
Soundex Match for *First Name, Last Name, Street name, Street Type, City, State, Country* individually
6c
All the matching processes take place within individual match
LookUp Match for *Street Type*
6d
Obtain match routine results for each attribute
7
Multiply aggregate match scores in Step 8 for each attribute with attribute weights. Sum up the results to obtain aggregate match score for the records and express in percent
9
For each attribute, multiply match routine result with each match routine weight. Sum up the results to have aggregate match score for each attribute and express in percent
8

Figure 3 **The Matching Process**

Note that city and state information are parsed separately as {city, state, country, zip} = {'Tucson', 'AZ', 'USA', '85705'} and are omitted from the address line.

Let's look at the steps needed to parse the address line:

1. Define an exhaustive lookup for street type and apartment type.
2. Define an exhaustive country or region-specific lookup for state. State lookup should be defined as such that AZ and Arizona are identified as the same (but incorrect spelling is not considered).
3. Concatenate Address 1 and Address 2 into a single address Line.
4. Search the address line string from the right-hand side for valid city, state name or ZIP code with respect to corresponding lookups. If true, then proceed to Step 5; otherwise proceed to Step 6.
5. Remove city, state or ZIP code information if found in the address line string.
6. Identify street and apartment tokens using the structure Street {[Number] [Name] [Type]}, Apartment {[Type] [Number]}. Token identification is based on searching [Type] with respect to values in street type and apartment type lookup, respectively.
7. Consider any remaining string snippet after Step 6 to be part of a location descriptor.

The implementation of this algorithm is shown in **Figure 6**. In this code I've used while loops to scan through the state, city and ZIP lookup tables for finding a match. The main function ParseAddress uses this to eliminate city, state and ZIP information from the address line.

Matching and Deduplication

The process of matching and deduplication in Step 3 (see **Figure 3**) starts with the identification of the following three types of attributes:

- Identity attributes, including Cust_ID, name (First_Name, Last_Name), address (House_no., Street_Name, City, State, Zip_Code, Country), Mob_Phone and so on.
- Discriminating attributes such as date of birth (DOB).
- Record-qualifying attributes such as country, region or postcode.

The application will prompt the user to select these attributes. At least one each of identity, discriminating and record-qualifying attributes need to be selected. Note that if any records are flagged due to inconsistencies or errors in the identity and discriminating attributes in the cleansing and standardization process, these records should not be considered for matching purposes.

As mentioned previously, matching will be performed based on four routines: absolute match, partial match, Soundex and lookup match. The program would prompt the user to select which routines will be used for each attribute. At least one routine should be selected for each attribute, though all four can be selected for an attribute.

Appropriate weights need to be assigned to each identity attribute depending on their importance to the business process—for example, identifying a customer (Step 4). Similarly, weights need to be assigned to each routine for each attribute. The program should prompt the user to define the weights on a scale of 1 to 5.

Finally, we get to Step 5 and start to perform matching based on discriminating attributes to obtain the match windows. Thus, if DOB is defined as the discriminating attribute, then separate windows need to be formed based on the DOB. This means that records within the same window would have the same value of DOB. In the subsequent steps, the matching process will be performed within the individual windows and not within records across different windows.

Absolute Match

In Step 6 I perform an absolute match for all attributes. This routine compares two fields and looks for an exact match only. A score of 100 is assigned for an exact match. Any other result is scored 0.

For example, if Field 1 contains "John" and Field 2 contains "John," it's an exact

Figure 4 **Name-Parsing Example**

Name	M	r	.		J	o	h	n
Position	1	2	3	4	5	6	7	8

Figure 5 **Address-Parsing Example**

House No.	Street Name	Street Type	Street Type	Apartment Number	Location Descriptor
421	East Drachman	Street	Suite	5A	beside McDonald's

match and is given a score of 100. If Field 1 contains "Lisa" and Field 2 contains "Laura," it's not an exact match and is given a score of 0.

The implementation of this absolute match algorithm is:

```
let AbsoluteMatch (attrOfFirstRec : string) (attrOfSecondRec : string) =
  let attrRec01 = attrOfFirstRec.Trim().ToLower()
  let attrRec02 = attrOfSecondRec.Trim().ToLower()
  match attrRec01, attrRec02 with
    | "", "" -> 100
    | _, _ -> if attrRec01 = attrRec02 then 100 else 0
```

Partial Match

Next, I can perform a partial match routine for all attributes. The partial match is used to determine the relationship to a blank value. This is useful when some records have, for example, the customer first name but not the customer last name, or vice versa. Sometimes a field is blank in both records. The partial match algorithm takes care of matching those records where one of the important fields might have been left blank.

Blanks and zeros are considered to be the same value. As for the absolute match, an exact match is scored 100. A blank field value versus a non-blank field value is scored 75, while a blank field value versus a blank field value is scored 65. Any other result is scored 0.

The implementation of the partial match algorithm is:

```
let PartialMatch (attrOfFirstRec : string) (attrOfSecondRec : string) =
  let attrRec01 = attrOfFirstRec.Trim().ToLower()
  let attrRec02 = attrOfSecondRec.Trim().ToLower()
  match attrRec01, attrRec02 with
    | "", "" -> 65
    | _, "" | "", _-> 75
    | _, _ -> if attrRec01 = attrRec02 then 100 else 0
```

Notice that this code is similar to the absolute match.

Figure 6 **Parsing Address Lines**

```
let MatchCityStateZip (addressPart : string) =
  // Match with a state
  let stateMatchFound = stateNameTable |>
    List.exists (fun (part1,part2) ->
    part1 = addressPart || part2 = addressPart)
  // Match with a city
  let cityMatchFound = cities |> List.exists (fun city ->
    city = addressPart)
  // Match with a ZIP code
  let zipMatchFound = zipCodes |> List.exists (fun zipCode ->
    zipCode = addressPart)
  stateMatchFound || cityMatchFound || zipMatchFound

// The main parsing address algorithm is as follows:
let ParseAddress (strAddress : string) =
  let mutable finalAddress = strAddress
  // Split the address
  let addressParts = strAddress.Split([|","|],
    StringSplitOptions.None)
  // Remove city, state and ZIP information from the address
  for i = 0 to addressParts.Length - 1 do
    let currPart = addressParts.[i].Trim()
    if MatchCityStateZip currPart then
      // Index of current address part in original string
      let currIndex = finalAddress.IndexOf currPart
      // Remove city, state, ZIP information along with the
      // following whitespace or comma
      finalAddress <- finalAddress.Remove(currIndex, currPart.Length)
  let finalAddress = finalAddress.Replace(", ,",",")
  let finalAddress = finalAddress.TrimEnd([|',';' '|])
  finalAddress
```

Soundex Match

Now I perform the Soundex algorithm for the attributes first name, last name, street name, city, state and country individually. The Soundex algorithm detects similar-sounding words using the following algorithm:

1. Capitalize all letters in the string.
2. Retain the first letter of the string.
3. After the first position, convert all occurrences of the following letters to blank: A, E, I, O, U, W, Y.
4. Change letters from the predetermined sets to corresponding number as shown in **Figure 7**.
5. Remove all consecutive pairs of duplicate digits and blanks from the string that resulted after Step 4.
6. Return the first four characters of the string, padded with trailing zeros if needed.

Scoring values for the Soundex routine are a bit different. As before, if both strings are equal they're scored 100. If one string is blank and the other is non-blank, then they're scored 50. If both strings are blank they're scored 0. And if neither string is blank and they're not equal, they're scored 0.

The implementation of this algorithm in F# is shown in **Figure 8**.

In this code the Soundex conversion is done using pattern matching, keeping the first character intact. The following two for loops find consecutive duplicate characters and replace the second such character with a blank. The next two for loops discard the blanks, effectively removing any duplicates. Thus the four for loops discard juxtaposed duplicate characters.

The following two if statements extract the first four characters and, if needed, pad with zeros to make it at least four characters. The final pattern matching implements the scoring for the Soundex routine.

Lookup Matching

Lastly, I perform a lookup match for the street type attribute. A street lookup table will be referenced to standardize the street type, like so:

```
let LookupMatch (streetName : string) =
  let mutable streetMatchFound = false
  let mutable i = 0
  while ((no streetMatchFound) && (i < streetNames.Length)) do
    if (streetName = streetNames.[i]) then
      streetMatchFound <- true
  match streetMatchFound with
  | true -> 100
  | false -> 0
```

Figure 7 **Soundex Letter Conversion**

Letter	Corresponding Number
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

Figure 8 **Soundex Match**

```
let SoundexMatch (attr1:string, attr2:string) =
  let conv c =
    match c with
    | 'A' | 'E' | 'I' | 'O' | 'U' | 'W' | 'Y' -> ' '
    | 'B' | 'F' | 'P' | 'V' -> '1'
    | 'C' | 'G' | 'J' | 'K' | 'Q' | 'S' | 'X' | 'Z' -> '2'
    | 'D' | 'T' -> '3'
    | 'L' -> '4'
    | 'M' | 'N' -> '5'
    | 'R' -> '6'
    | _ -> c

  let convertSoundex (inp:string) =
    // Capitalize all letters in the string
    let chars = inp.ToUpper().ToCharArray()
    let chars =
      [| // Retain the first letter of the string
      yield chars.[0]
      // Keep the first character, and remove pairwise-duplicates
      // Change letters from the predetermined sets to
      // corresponding number
      for (c1,c2) in Seq.pairwise (Seq.map conv chars) do
        // Remove all consecutive pairs of duplicate digits
        // and blanks from the string
        if c1 <> c2 && c2 <> ' ' then yield c2 |]
    // Convert back to a string
    String chars

  // Retain first four characters of resultant strings padded
  // with trailing zeros if needed; leave unchanged if any
  // string is blank
  let adjustResult (result:string) =
    match result.Length with
    | n when n >= 4 -> result.Substring(0, 4)
    | 0 -> result
    | n -> result + String.replicate (4 - n) "0"

  let attr1Result = attr1 |> convertSoundex |> adjustResult
  let attr2Result = attr2 |> convertSoundex |> adjustResult

  match attr1Result, attr2Result with
  | "", "" -> 0
  | "", _ | _, "" -> 50
  | _, _ -> if (attr1Result = attr2Result) then 100 else 0
```

This code scans through the street lookup table to find a street name match using a while loop and then returns the scores if a match is found.

Scoring the Results

In Step 7 of the matching process, I retrieve the scores of the matching processes for each attribute. Thus, for first name, if there is no match for the absolute match routine, but there is a match for the Soundex routine, the match routine scores for first name would be 0 and 100, respectively.

Figure 9 **Aggregated Scoring**

```
let WeightedAverage results =
  let cumulativeWeight = results |>
    Array.sumBy (fun (r, weight) -> r * weight)
  let totalWeight = results |>
    Array.sumBy (fun (r, weight) -> weight)
  cumulativeWeight / totalWeight

// Aggregate match score
// Calling the match routine which in turn calls absolute match,
// Partial Match and Soundex Match routines in parallel
let FindAggregateMatchScore row =
  let resultsWithWeights =
    Async.Parallel [ async { return AbsoluteMatch row, 5 }
                     async { return PartialMatch row, 5 }
                     async { return SoundexMatch row, 4} ]
    |> Async.RunSynchronously

  WeightedAverage resultsWithWeights
```

Figure 10 **Weighted Scoring**

```
// Percentage match score
let FindPercentageMatchScore(rows : seq<string * string>) =
  let results =
    Async.Parallel [ for row in rows ->
    async { return FindAggregateMatchScore row } ]
    |> Async.RunSynchronously

  // Apply a weight of 5 for the first attribute and a weight
  // of 4 for second and a weight of 3 for all other attributes
  let resultsWithWeights = results |>
    Array.mapi (fun i r ->
    r, (match i with 0 -> 5 | 1 -> 4 | _ -> 3))

  WeightedAverage resultsWithWeights
```

In Step 8, the weighted match score for each attribute is determined, giving the attribute a score from 1 to 5. For example, if the absolute matching score of 0 is weighted as 5, and the Soundex score of 100 is weighted as 4, then the aggregate score for first name would be:

[(0x5)+(100x4)]/(5+4) ~ 44%

Assuming that all the match algorithms are selected, the implementation of this weighted scoring is shown in **Figure 9**.

This code assumes that all three match routines are called for each attribute (though not for the street attribute, for which the lookup match should be performed). First, Func delegates are declared for each match routine. Then the delegates are invoked in an asynchronous fashion using the BeginInvoke method. After waiting for the tasks to complete via WaitHandle.WaitAll, the results are collected using EndInvoke methods that take IAsyncResult parameters returned during BeginInvoke calls. Finally, the aggregate match score is returned as the last expression in the function.

In Step 9, the aggregate match scores of each attribute are multiplied by individual attribute weights, then added and expressed as a percent match score between the two records (see **Figure 10**).

The Task.Factory.StartNew method from the Task Parallel Library for F# is used to call the aggregate match score for each pair of attributes of the two rows of data being compared, followed by a for loop that calculates the cumulative result. Finally, the percentage match score is returned.

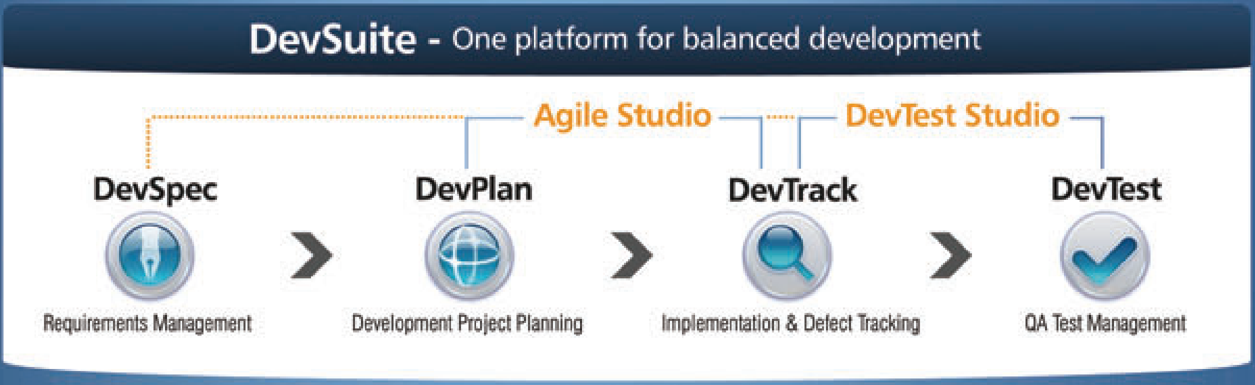
Match thresholds—the upper threshold and lower threshold for scores—are user-defined. The system will ask the user for defining the upper and lower thresholds. A record match score above the upper threshold is considered an automatic match, while a record match score below the lower threshold is rejected and is considered a new customer record. A score between and inclusive of these two thresholds should be flagged for review.

With that, you've completed the record-matching and deduplication process. Obvious duplications can be deleted, and you can pass on the records flagged for review to either a real human or a more extensive programmatic review process. ■

AMBAR RAY *is a solution architect working on Microsoft technologies. Ray has nearly 12 years of experience in the IT industry.*

THANKS to the following technical experts for reviewing this article: *Don Syme, Technical Excellence Group of TechMahindra Ltd.*

TechExcel
Application Lifecycle Management
Application Lifecycle Management
Application Lifecycle Management
Application Lifecycle Management



Managing Employee Rewards with Office and SharePoint BCS

Ying Xiong

Managing employee rewards is a business-critical function at all companies. This is especially true for large enterprises such as Microsoft. At Microsoft, we offer many types of rewards to eligible employees, such as merit-based pay increases, promotions, bonuses, stock and other awards. Managing these rewards according to company guidelines and budgets (set for each region, business unit or organization, employee salary plan, and pay level) is a complex process.

Many business rules are involved when giving rewards to employees. Among other goals, these rules ensure that our top performing employees at all levels receive highest rewards while the organizations at all levels meet their goals and remain within their budgets and guidelines. Every year, managers and HR personnel analyze the numbers and determine each reward's guidelines and limits at all levels. This process (called calibration) is time-consuming and complicated.

This article discusses:

- BCS basics
- Defining ECTs
- Creating an Excel add-in
- Using the BCS APIs

Technologies discussed:

Office 2010, SharePoint 2010, Business Connectivity Services

The current rewards-management tool used by our managers and HR personnel is a Windows Forms application. The tool works as designed, but it doesn't meet all of our needs—there are a number of ways it could be improved to simplify and streamline the calibration process.

In response to suggestions for improving the calibration tools, the Microsoft IT (MSIT), HR Business and Microsoft Office and SharePoint product teams collaborated to create a new solution based on Microsoft Business Connectivity Services (BCS), a feature of Microsoft Office 2010 and SharePoint 2010. BCS provides read/write access to external data from line-of-business (LOB) systems, Web services and databases, as well as SharePoint and Office applications.

The new solution uses Microsoft Excel 2010 as the rewards-management UI, and leverages BCS technology to cache and synchronize employee data and business rules between the user's local machine and the back-end systems (Rewards Web Services and a SQL Server database).

This article shares the experiences of the Microsoft teams in developing and deploying the rewards-management BCS solution.

Where We Started

The existing rewards-management tool was designed to support an integrated performance management experience for overall employee assessment, calibration, ratings and rewards through a single cohesive toolset. **Figure 1** shows the architectural design of the existing tool. This tool is a typical three-tier application where the UI layer is a Windows Forms application that reads data from and writes data to back-end Web services. These services query and update employee records, business rules and other reference data in a SQL Server database.

The solution was designed to minimize network traffic between users' computers and Web servers hosting the Web services component. When the client app starts, it retrieves and caches all necessary reference data, business rule data and employee records in the organization to which the user has access permissions. Most of the rewards-management business logic and rules are implemented in the client component. Hence, when the user assigns or changes employee rewards, the corresponding business rules are fired to validate the changes without calling Web services at the back end.

The Web services component was built with the Microsoft .NET Framework 2.0 Web services framework (ASP.NET Web Services). As mentioned earlier, it returns employee records and business rules data to the client application and saves data changes passed from clients to the database. There's business validation logic in the Web service to ensure data integrality before the data is saved to the database. All Web services calls are designed as synchronous methods.

Users get error messages when Web services calls fail, and users can take action based on the nature of the error. In addition, the Web service was designed to serialize employee and other data records into a byte array and return the byte array to clients in order to minimize the size of data transferred between Web services servers and the client app.

Most of the rewards-management business logic and rules are implemented in the client component.

Finally, the rewards data was stored in a relational database implemented in Microsoft SQL Server 2005. This database stores all data needed for managing employee rewards, as well as historical data for past performance-review periods.

As I mentioned previously, this application architecture worked largely as planned and was a significant improvement over previous rewards and calibration systems. However, in use, we discovered many areas where we could improve the tool to make calibration easier and less time-consuming. So that you'll understand the reasoning behind design decisions for the new service, let's take a quick look at some of the issues discovered in the old one.

The first, and probably most important, issue encountered in the old calibration tool involved the difficulty of modeling rewards. During the calibration process, managers need to balance two variables. Each employee must receive appropriate rewards based on a performance rating. However, the entire organization needs to meet a predefined rewards budget. Getting this balance right is a time-consuming process. Managers need to enter employee rewards numbers, see how they affect the team rewards budget, and adjust rewards to balance both reward rules and budget. This can take multiple iterations before the rewards are finalized.

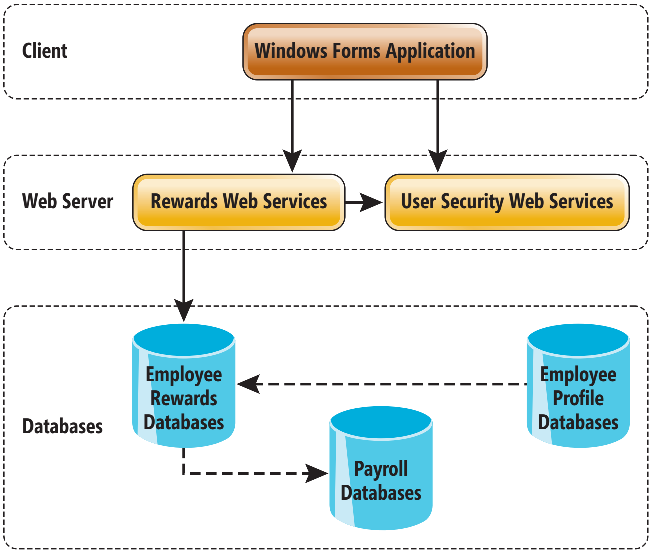

Client
Windows Forms Application
Web Server
Rewards Web Services
User Security Web Services
Databases
Employee Rewards Databases
Employee Profile Databases
Payroll Databases

Figure 1 **Architecture of Current Rewards-Management Tool**

The existing tool didn't facilitate the rewards-versus-budget-balancing process. It didn't provide capabilities for managers to enter different numbers for each reward and analyze the result to see the impact to the budgets and guidelines. Instead, managers often had to export the employee data into an Excel spreadsheet and manually model rewards and budget figures.

When calibration is complete and each employee's rewards are finalized, managers have to enter the data into the tool manually and submit the rewards for approval. This double data entry is also a time-consuming process.

The Windows Forms-based tool requires users to be online and on the corporate network because the tool needs access to back-end Web services for data reads and writes. This adds another level of hassle for employees with busy schedules or who travel frequently.

Finally, while the existing rewards-management tool provided a built-in set of reports, we had many requests for additional reports and flexible ad hoc reporting capabilities to make the calibration process more transparent and efficient.

BCS Basics

BCS lets you use Office applications (Excel, Outlook, InfoPath, Word and so on) and SharePoint to process data from back-end systems. As I already mentioned, BCS is a component for both Office 2010 and SharePoint 2010, so it's available on both client and server machines. (See msdn.microsoft.com/library/ee556826 for more information about the BCS tooling and runtime components.)

When designing a BCS-enabled solution, you start by defining an entity model that can connect to external systems and map data from those external data structures into the BCS data structure. This entity model is called an External Content Type (ECT). You can create as many ECTs as you want for your solution through SharePoint Designer or Visual Studio.

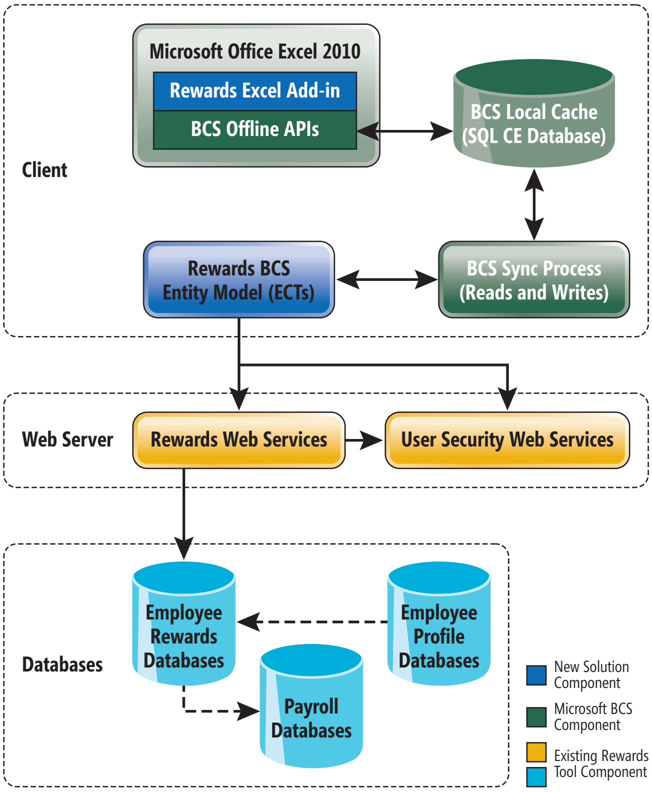

Microsoft Office Excel 2010
Rewards Excel Add-in
BCS Offline APIs
BCS Local Cache (SQL CE Database)
Client
Rewards BCS Entity Model (ECTs)
BCS Sync Process (Reads and Writes)
Web Server
Rewards Web Services
User Security Web Services
Databases
Employee Rewards Databases
Employee Profile Databases
Payroll Databases
New Solution Component
Microsoft BCS Component
Existing Rewards Tool Component

Figure 2. **Architecture of the New Rewards Solution**

ECT metadata, including the code for defining the entity model, is published and stored in a SharePoint metadata store. The published ECT model can be downloaded and installed on user client machines through a SharePoint workspace or via a standalone packaging tool provided by BCS.

At run time, the BCS runtime component on either the client or on the server invokes the ECT model to communicate with external systems. On the client, the data from any external systems is cached on the user's machine in a SQL Server Compact Edition (SQL CE) database. The cached data can then be displayed and manipulated in Office applications through the BCS APIs. The changes made to the cached data by users will be queued on users' machines in the same SQL CE database. The BCS runtime component is then responsible for updating the changes to external systems through the ECT model.

The New Rewards Tool

For the new rewards-management tool, we created a solution using the BCS framework, with Excel as the rewards-management UI, to address business issues described earlier. In this Excel/BCS solution, we defined a total of 15 ECTs for employee information, business rules and other reference data. The BCS runtime uses these ECTs to connect to existing rewards-management Web services (as external systems) and to retrieve and cache employee and business rules data.

The employee data is rendered as an Excel worksheet through an Excel add-in that uses BCS local cache APIs. Managers can assign, calibrate and manage their employee rewards completely within Excel and still be able to enforce all business rules based on rule data cached in the BCS local store—even if the managers are offline. With the Excel and BCS solution, managers are able to see a list of selected employees (in the Excel worksheet) and various statistics (in the Excel Task panel) on one screen. The statistics at the bottom are dynamically updated as reward values change, and different statistics are displayed depending on the field selected by the user. This is a greatly improved user experience.

Using native Excel functionality, such as pivot tables and charts, managers and HR personnel can create and visualize various reports for the data cached by BCS, thus improving productivity. Through Excel templates, users can create reports more easily, and the reports don't require IT development.

Aside from business benefits, an interesting technical point we discovered while developing this new rewards-management solution was that we could leverage the existing Rewards Web Services component in the BCS solution even though the Web services weren't designed or optimized for BCS. We were also able to leverage existing business rule data objects and validation code within our Excel add-in solution. As a result, we are able to deliver and demo the new solution in a relatively short period of time.

Solution Architecture

The new solution architecture is shown in **Figure 2**. As you can see, the changes to the original architecture were made only for the client architectural component. In the diagram, the green boxes are BCS components that are already installed on users' machines when they install Office 2010. The blue boxes are the components developed for the new solution, and the light-blue boxes are components of the existing rewards-management tool.

When the ECTs developed for the new rewards solution are installed on a user machine, the BCS sync process (BCSSync.exe) immediately invokes the entity model to retrieve data for each ECT defined in the model from the external rewards Web services. The actual data synced is determined based on the user's data-access permissions (obtained via user security Web services). This step requires that the user be online in order to access Web services.

BCS lets you use Office applications and SharePoint to process data from back-end systems.

Retrieved data such as employee records for an organization is stored in a SQL CE database created by the BCS sync process as the local data cache for the user. The data in SQL CE is encrypted using the user's certificate and the data is secure at the user level.

The BCS sync process runs periodically to synchronize the user's local cache with the rewards server for any changes made on the server side. The BCS framework enables the ability to define synchronization frequency for each ECT differently. This

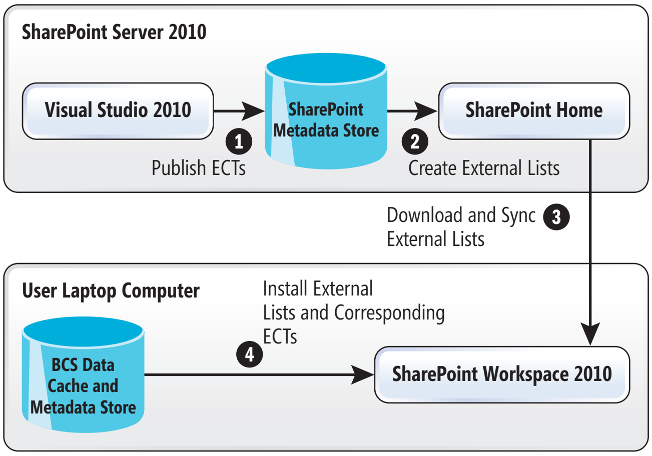

SharePoint Server 2010
Visual Studio 2010
SharePoint Metadata Store
SharePoint Home
1
Publish ECTs
2
Create External Lists
Download and Sync External Lists
3
User Laptop Computer
Install External Lists and Corresponding ECTs
4
BCS Data Cache and Metadata Store
SharePoint Workspace 2010

Figure 3 **ECT Deployment Process Through SharePoint Workspace**

means we can define a longer (hours or days) sync frequency for infrequently changed entities such as business rules, and shorter (minutes or hours) sync frequency for employee rewards data that can change frequently during the review period.

With employee, business rules and other reference data cached on the user's machine, the user can start the rewards-management tool by simply launching Excel. The rewards Excel add-in uses BCS APIs to retrieve data from the SQL CE database and binds the data into an Excel worksheet. From that point the user can see the employee records and their rewards data in Excel. This Excel file is no different than other Excel files you normally create or use, and the user can employ native Excel functions to manipulate and analyze the data.

When the user makes changes in the worksheet to assign or update employee rewards such as salary merit increase percentage, promotion amount or stock award shares, corresponding business rules are fired to validate and process these changes. We're able to achieve this by hooking the value change event of an Excel cell to existing business rule code.

Moreover, some employee data such as employee number, job title and e-mail address can't be changed by users. We accomplish this requirement by using a native Excel lock feature.

When the user is ready (online or offline) to submit changes to the back-end server, the changes are submitted into a local queue stored in the SQL CE database. The BCS sync process picks up changes one at a time from the queue and sends the changes to the back-end Rewards Web Services. If the user is offline or the back-end Web services are not available, the BCS process will retry the change until the user is online or the Web services are available.

If the user isn't ready to submit the changes to the back-end server and wants to save changes as a draft, this is handled as a normal Excel file save operation.

Defining ECTs

As stated previously, an important first step for any BCS-enabled project is to create the ECTs for your solution. Designing ECTs is the process of modeling the data entities you'll use in your Office or SharePoint solution. The number of ECTs and the data structure for each ECT depends on the nature of your application data, how the data is used in your application and the interfaces provided by your external systems.

Because the data for ECT entities come from external systems (the Rewards Web Services, in our case), the simplest way to model your entities is to design them the same as the returned types from your external systems. This involves no data mapping from one structure to another structure. However, you can't always do this, especially when external Web services return data types that are complex and have a deep hierarchy. This was the case for the rewards-management system.

The existing Rewards Web Services return a serialized byte array (for employee and business rule data) that needs to be deserialized on the client side into custom System.Data.DataTable types. These custom data types aren't supported by default in BCS.

Our ECT implementation for the new rewards system defines a relatively flat data structure (with simple data types) for employee and business rule entities. It also converts or maps the data from the custom data tables (after deserialization) into the flat ECT structure. When users update data through ECTs, we then convert data back to custom data table types and send the changes to the back-end Web services.

Every ECT must have at least two methods defined. The Finder method is invoked by the BCS runtime to download all data. In the case of the employee ECT, it returns all employee records the user has permission to access. When an ECT is installed on a client machine, the installation process will create a subscription based on the Finder method for the BCS runtime to periodically synchronize the ECT.

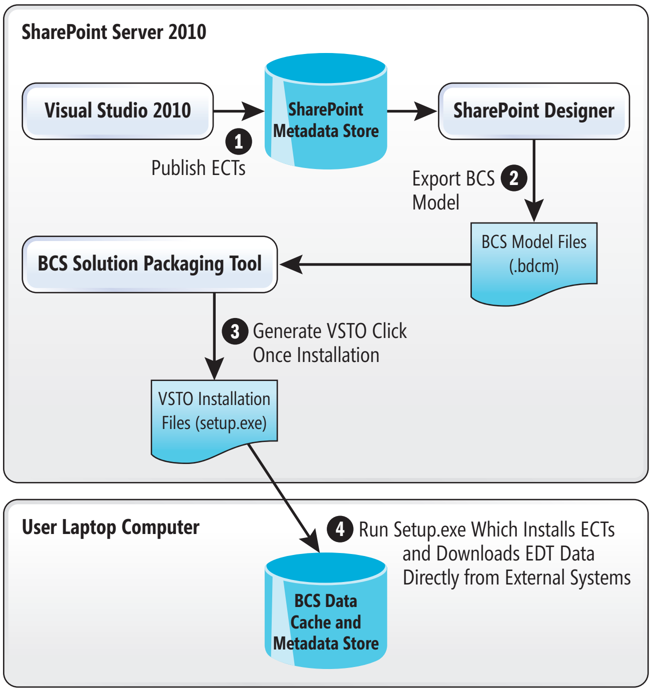
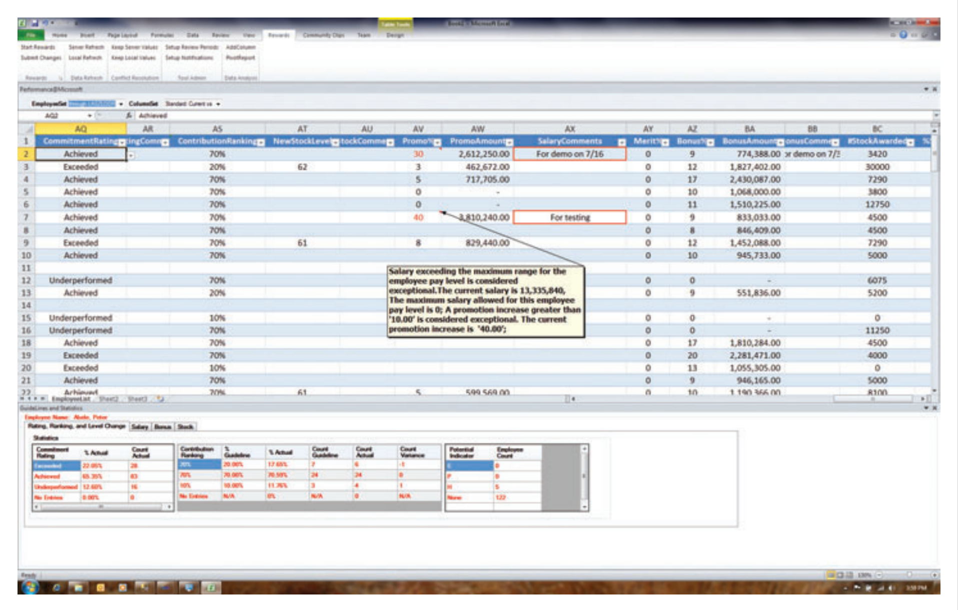

SharePoint Server 2010
Visual Studio 2010
SharePoint Metadata Store
SharePoint Designer
1
Publish ECTs
Export BCS Model
2
BCS Model Files (.bdcm)
BCS Solution Packaging Tool
3
Generate VSTO Click Once Installation
VSTO Installation Files (setup.exe)
User Laptop Computer
4
Run Setup.exe Which Installs ECTs and Downloads EDT Data Directly from External Systems
BCS Data Cache and Metadata Store

Figure 4 **ECT Deployment Process Through BCS Solution Packaging Tool**



AQ2 Achieved

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC
1	CommitmentRating	ingComm	ContributionRanking	NewStockLevel	tockComme	Promo%	PromoAmount	SalaryComments	Merit%	Bonus%	BonusAmount	onusComme	#StockAwarded
2	Achieved		70%			30	2,612,250.00	For demo on 7/16	0	9	774,388.00	or demo on 7/1	3420
3	Exceeded		20%	62		3	462,672.00		0	12	1,827,402.00		30000
4	Achieved		70%			5	717,705.00		0	17	2,430,087.00		7290
5	Achieved		70%			0	-		0	10	1,068,000.00		3800
6	Achieved		70%			0	-		0	11	1,510,225.00		12750
7	Achieved		70%			40	3,810,240.00	For testing	0	9	833,033.00		4500
8	Achieved		70%						0	8	846,409.00		4500
9	Exceeded		70%	61		8	829,440.00		0	12	1,452,088.00		7290
10	Achieved		70%						0	10	945,733.00		5000
11													
12	Underperformed		70%						0	0	-		6075
13	Achieved		20%						0	9	551,836.00		5200
14													
15	Underperformed		10%						0	0	-		0
16	Underperformed		70%						0	0	-		11250
18	Achieved		70%						0	17	1,810,284.00		4500
19	Exceeded		70%						0	20	2,281,471.00		4000
20	Exceeded		10%						0	13	1,055,305.00		0
21	Achieved		70%						0	9	946,165.00		5000
22	Achieved		70%	61		5	599,569.00		0	10	1,190,366.00		8100

Salary exceeding the maximum range for the employee pay level is considered exceptional.The current salary is 13,335,840, The maximum salary allowed for this employee pay level is 0; A promotion increase greater than '10.00' is considered exceptional. The current promotion increase is '40.00';

Commitment Rating	% Actual	Count Actual
Exceeded	22.05%	28
Achieved	65.35%	83
Underperformed	12.60%	16
No Entries	0.00%	0

Contribution Ranking	% Guideline	% Actual	Count Guideline	Count Actual	Count Variance
20%	20.00%	17.65%	7	6	-1
70%	70.00%	70.59%	24	24	0
10%	10.00%	11.76%	3	4	1
No Entries	N/A	0%	N/A	0	N/A

Potential Indicator	Employee Count
E	0
P	0
H	5
None	122

Figure 5 **Excel Design for the Rewards-Management Solution**

The SpecificFinder method returns the specific instance of data (employee) based on the entity identifier (employee number). SpecificFinder is used by the BCS runtime to synchronize the employee instance with the back-end server. For data entities that users update on the client side, such as employee records, the ECTs include an update method that the BCS runtime can invoke to send updates to the server. Similarly, if users need to create new records of an entity, a create method is defined in the ECT. Reference data entities that users won't change don't need the update or create methods for their ECTs.

In your own ECT implementations, you can add any business logic you need to validate or process data before invoking external Web services. This is a great design option with BCS that lets you put business logic right in your ECTs and run it on both the client and the server.

There are two ways you can design and create ECTs. If your external systems return relatively flat and strongly typed data types, you can use SharePoint Designer to generate your ECTs. SharePoint Designer generates ECTs based on external system interfaces—operation and data contracts in the case of Windows Communication Foundation (WCF) Web services. However, at this time, SharePoint Designer doesn't support complex and custom types such as custom data sets and data tables.

The second and more powerful approach is to use Visual Studio to design and create ECTs. Visual Studio 2010 includes a project template specifically for creating BCS entity models. This is the approach we used in our rewards solution.

ECT Deployment and Installation

To make the rewards-management system work, the ECT models need to be downloaded and installed on the client machine. Microsoft Office and SharePoint 2010 provide two ways for deploying ECTs. **Figure 3** shows a deployment process using SharePoint Workspace, which is installed on users' computers as part of the Office 2010 installation. **Figure 4** illustrates how it works.

First, ECTs are published from Visual Studio to the SharePoint Server metadata store (step 1). For Visual Studio to publish ECT metadata into the SharePoint data store, it must run on the same server where SharePoint 2010 is installed.

Next, you create an external list for each published ECT (step 2). You can create the external list through the SharePoint homepage from any computer that can connect to SharePoint Server.

After you create the external list for an ECT, SharePoint immediately runs the ECT, which connects to and retrieves data from the external systems and displays the data as list content on the SharePoint site. With this deployment process, you can view the data returned from your ECT on the SharePoint Server, in addition to your office application on the client. In some situations this is a desirable feature, but it's not required. The rewards-management solution doesn't use this feature because we don't want sensitive employee-rewards data to be shown through SharePoint.

Now download the external lists created from SharePoint Server in step 2 to your local machine (step 3). You can start the downloading process by connecting to the SharePoint homepage from Internet Explorer and selecting Site Actions | Sync to SharePoint Workspace. This will launch the SharePoint Workspace program on your client machine and start downloading external lists one at a time. With the current release of Workspace 2010, the user will have to click the Install button for each external list to be downloaded.

An important first step for any BCS-enabled project is to create the ECTs for your solution.

By downloading an external list, SharePoint Workspace will automatically install the ECT (metadata and assembly) into the BCS local data store (step 4). In addition, a subscription is created for the installed ECT for BCS runtime to synchronize ECT data periodically with the external systems. The default data sync frequency is six hours.

The rewards-management solution uses a second deployment process to install ECTs onto users' client machines (as shown in **Figure 4**).

In this deployment method we use the BCS Solution Packaging Tool (code.msdn.microsoft.com/odcsps14bcspkgtool) and there are no

external lists involved in the deployment process. Therefore, there's no need for SharePoint homepage and SharePoint Workspace, though we still employ the SharePoint Server metadata store.

In this deployment process, after the ECTs are published into SharePoint metadata store, we use SharePoint Designer to export the ECTs into a model file (.bdcm). You can run SharePoint Designer either on the server machine or on your client machine and you can place the exported model file in a shared folder. The BCS Solution Packaging Tool reads the model file and generates a Visual Studio Tools for Office (VSTO) installation package that can be run by users.

From the user's perspective, this is a simple, easy and transparent installation process. Users only click once when running the setup.exe to install all ECTs exported from SharePoint Designer. The installation will also immediately start BCSSync.exe to run the ECTs and download ECT data directly from external systems into the BCS local data cache.

The Excel Add-In

As described earlier, we redesigned the existing Windows Forms application component into an Excel add-in component and used Excel as the UI for managing employee rewards. The Excel add-in performs the following main functions:

- Renders employee records and related guidance and statistics as an Excel worksheet and task pane.
- Uses BCS APIs to retrieve employee and business rule data from the BCS local cache, and writes back any changes to the BCS local update queue.
- Invokes existing business logic code when users make changes to employee rewards to ensure the changes meet corresponding guidelines and minimum/maximum ranges.

Figure 5 shows the new Excel-based UI for rewards management. The top half of the screen is an Excel worksheet displaying employee records for the users' organization. The bottom half of the screen is a task pane that shows the guidelines for the highlighted employee (selected row) and statistics for the organization. Guidelines and statistics for selected columns will be displayed on the bottom screen.

Figure 6 **Retrieving Employee Records**

```
string entityNameSpace = "MSIT.HR.Rewards.BCS";
string entityName = "EmployeeDetails";

// Initialize and connect to BCS local cache
RemoteSharedFileBackedMetadataCatalog catalog =
  new RemoteSharedFileBackedMetadataCatalog();

// Get the ECT for employee entity
IEntity entity = catalog.GetEntity(@entityNameSpace, entityName);

// Get the finder method defined for employee ECT
string methodName = entity.GetMethodInstance(
  MethodInstanceType.Finder)[0].Value.Name;
IMethodInstance mi = entity.GetMethodInstance(
  methodName, MethodInstanceType.Finder);

// Get external system instance
ILobSystemInstance LobSysteminstance =
  entity.GetLobSystem().GetLobSystemInstances()[0].Value;

// Retrieves all instances of employee ECT
IEntityInstanceEnumerator instanceEnumerator =
  entity.FindFiltered(entity.GetDefaultFinderFilters(),
  mi, LobSysteminstance, OperationMode.CachedWithoutRefresh);

// Loop each employee and add it to Excel worksheet list object
while (instanceEnumerator.MoveNext()) {
  ... // Loop each instance of employee record and
      // add it to Excel worksheet list object
}
```

This simple and easy-to-use Excel interface helps managers and HR personnel effectively manage their employee rewards. They'll perform all their work on one screen and still see the overall rewards picture for their organizations. Of course, they can build additional statistics and charts based on worksheet employee data using native Excel functions such as pivot tables and pivot charts.

When the Excel program is started, there's a custom Rewards tab added by the Excel add-in in the menu bar. Users click the Rewards tab and a ribbon is shown with various buttons for users to start and use the rewards-management solution.

> Each ECT has a unique entity name within an entity name space.

Using the BCS APIs

Of course, the BCS APIs lie at the center of any BCS-based solution. Let's walk through some of our sample code to illustrate the use of BCS APIs.

First, let's take a look at some code used to retrieve all employee records from the local cache store (see **Figure 6**).

IEntity is the main BCS interface for ECTs you define for your solution. IEntityInstance represents a specific data instance for the entity you want to get data from. Each ECT has a unique entity name within an entity name space. Each entity instance is identified by a unique ID you define when you define the ECT. In this case, it's the employee number.

You can update an employee record (or, more specifically, an employee ECT instance) through a BCS entity instance update method. This creates an update operation in the BCS local queue. Thus there will be 10 operations queued in local cache if you make changes to 10 employee records. BCS sync will process one operation at a time, like so:

```
// Query the employee instance you want to change
Identity identity = new Identity(employeeNumber);
IEntityInstance myEmployee =
  entity.FindSpecific(identity, LobSysteminstance);

// Update the employee bonus amount
myEmployee["BonusAmount"] = 1000;

// Submit the changes to BCS local cache
myEmployee.Update();
```

The code in **Figure 7** shows how to query the BCS cache for all pending or failed operations due to external system unavailable, error and data conflicts.

ISynchronizationManager is the main interface for the BCS entity-synchronization process. Each operation (IOperation) is a change to entity data to be processed by the BCS runtime. After processing, each data update operation will have a status of either Pending or In Error. If an operation is processed successfully, the

/update/2011/02

ComponentSource®
The Definitive Source of Software Components
www.componentsource.com

BEST SELLER

FusionCharts | from $195.02

InfoSoft Global
empowering human thoughts

Interactive Flash & JavaScript (HTML5) charts for web apps.

- Liven up your web applications using animated & data-driven charts
- Create AJAX-enabled charts with drill-down capabilities in minutes
- Export charts as images/PDF and data as CSV from charts itself
- Create gauges, dashboards, financial charts and over 550 maps
- Trusted by over 17,000 customers and 330,000 users in 110 countries

BEST SELLER

Spread for Windows Forms | from $959.04

GrapeCity PowerTools

A comprehensive Excel compatible spreadsheet component for Windows Forms applications.

- Speed development with Spreadsheet Designers, Quick-start Wizard and Chart Designers
- Automatic completion - provide "type ahead" from user input to cell
- Features built-in tool to design charts with 85 different styles
- Preserves Excel files and restores unsupported functionality
- Includes sleek new predefined skins and the ability to create custom skins

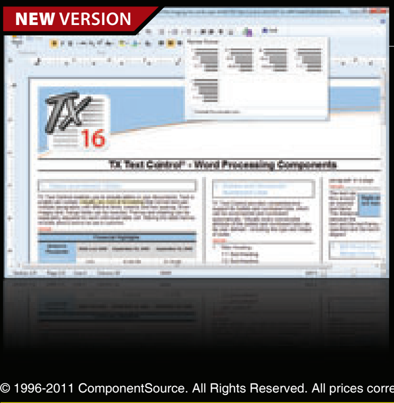
BEST SELLER

ActiveReports 6 | from $685.02

GrapeCity PowerTools

Latest release of the best selling royalty free .NET report writer.

- Now supports Windows Server 2008, 64Bit & IE8.0
- First Flash based Report Viewer for end users
- Supports PDF Digital Signatures, RSS Bar Codes and external stylesheets
- Features an easy-to-use Visual Studio report designer and a powerful API
- Offers seamless run-time deployment, royalty free


NEW VERSION

TX Text Control .NET for Windows Forms/WPF | from $499.59

TX TEXT CONTROL
word processing components

Word processing components for Visual Studio .NET.

- Add professional word processing to your applications
- Royalty-free Windows Forms and WPF rich text box
- True WYSIWYG, nested tables, text frames, headers and footers, images, bullets, structured numbered lists, zoom, dialog boxes, section breaks, page columns
- Load, save and edit DOCX, DOC, PDF, PDF/A, RTF, HTML, TXT and XML

We accept purchase orders.
Contact us to apply for a credit account.

US Headquarters
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
USA

European Headquarters
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
United Kingdom

Asia / Pacific Headquarters
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japan
102-0083

Sales Hotline - US & Canada:
(888) 850-9911
www.componentsource.com

VISA DISCOVER

GSA Schedule
Contract GS-35F-0168R

operation is removed from the queue and you won't be able to query the operation through the synchronization manager.

A Pending status indicates the operation is pending due to external system unavailability. BCS will automatically retry until it's successful or encounters an error.

An In Error status indicates that the operation failed to be processed by external systems due to a validation error, a data integrity error or a conflicting data error.

There will be a time when users want to explicitly refresh their local cache from external systems, instead of waiting for the BCS runtime to sync periodically. Here's how you force a refresh of data for all ECTs:

```
// Initialize and connect to offline cache
RemoteOfflineRuntime offlineRuntime = new RemoteOfflineRuntime();
ISubscriptionManager subManager =
  offlineRuntime.GetSubscriptionManager();

// Get all subscriptions
ISubscriptionCollection mysubs = subManager.GetSubscriptions();
IEnumerator<ISubscription> ie = mysubs.GetEnumerator();

while (ie.MoveNext()) {
  ISubscription mySub = ie.Current;
  mySub.RequestRefresh(true);
}
```

As mentioned earlier, the first time an ECT is deployed and installed on a user's machine, the BCS runtime creates a subscription to the data from external systems in its local data store. The subscription is being used by the BCS runtime to synchronize the ECT data with external systems.

In the sample code, ISubscriptionManager is the interface to get all subscriptions (ISubscription) created in the BCS data store. With a subscription object you can do a number of things, including changing the sync frequency for the ECT, adding additional query parameters, getting the last refresh status, requesting a refresh immediately (as in the example) and getting the number of instances synced for the ECT.

Figure 7 **Checking for Pending or Failed Operations**

```
// Initialize and connect to offline cache store
RemoteOfflineRuntime offlineRuntime = new RemoteOfflineRuntime();
ISynchronizationManager syncManager =
  offlineRuntime.GetSynchronizationManager();
IMetadataCatalog catalog =
  offlineRuntime.GetMetadataCatalog();

// Get all current operations from local operation queue
IOperationCollection allOps = syncManager.GetOperations();

foreach (IOperation op in allOps) {
  // ECT associated with the operation
  IEntity entity = op.Entity;

  // Query for operation status
  string operationstatus = op.OperationStatus.ToString();

  // Query for operation retry count
  int retryCount = op.RetryCount;

  // Get last exception message if operation errored out
  string errMsg = op.LastException.Message;

  // Retry the operation if status is pending or error
  op.Retry();

  // Decide if error is due to data conflict
  if (errMsg.Contains("ConflicDetectedException"))
    // Then the error is due to data conflict
}
```

Lessons Learned

During the design, development and deployment of the new rewards-management solution, we gained much practical, real-world experience with both Office add-in development and SharePoint 2010 BCS implementation. Here are some of the insights we gleaned along the way:

First, spend as much time as you can designing the ECTs for your solution. ECT design is key to a successful BCS solution. ECTs directly affect the behavior of BCS runtime, and a badly designed ECT can cause the BCS runtime to crash or throw out-of-memory exceptions.

We learned this lesson the hard way. We initially designed a big ECT that returned all business rules as a single instance. The single instance contained a 40MB array that caused the BCS runtime to peak at 600MB of memory use when the array was de-serialized to objects.

To mitigate memory-use problems, we then redesigned the big ECT into a few separate ECTs based on rule categories. Each category ECT returns a number of entity instances equal to the actual number of rules. With these new, more-focused ECTs, the BCS sync process works much more efficiently.

ECT design is key to a successful BCS solution.

Another best practice is to use strong typing and a relatively flat structure as much as possible to define your ECTs. If you can use SharePoint Designer to generate ECTs, it will save a lot of development time. Creating ECTs in Visual Studio with a lot of entity attributes gives you more flexibility, but it can be a time-consuming and tedious process.

Think through your requirements as to how much data can be downloaded and cached to a user's local machine. This determines how you'll implement the finder method for your ECT. Of course, the data returned from your external system calls sets the limit for the number of records you can download. If your ECT design needs more data than the limit, then the external systems will need to be changed to meet your requirement.

In our rewards solution, a manager can download and cache the employee records for his organization only. Thus, two different users are likely to have two different sets of employee data downloaded to their local cache. The existing Rewards Web Services authenticate the user and return only the employee data the user is allowed to access. In other words, the data filtering is done at the external systems and the ECT implementation doesn't have to filter the data based on user permissions. If this is not the case for your solution, you'll need to consider additional design and development time for filtering ECT data for the local cache.

In your ECT implementation, handle external system exceptions carefully. At some point, your ECT implementations for the Finder and SpecificFinder methods will need to invoke external system methods to retrieve data. You'll need to catch and handle exceptions thrown from external systems in your code. However, you'll need to throw the exception back to the BCS runtime after you handle it. Otherwise, the BCS runtime will think the call to the external systems was successful, return zero rows of data and remove the data already in the local cache. You definitely don't want the local data cache deleted because of an external system exception.

There are additional considerations regarding the use of third-party libraries in your ECT implementation. In one of our ECTs, we had to reference an assembly developed by another group to perform additional business logic and data transformation. During the deployment of the ECT, we found that the referenced assembly wasn't deployed to client machines as part of the ECT installation. As a result, BCS failed to run the ECT and download the data.

As I write this, we're working with the BCS product team to engineer a solution. The workaround is to add the assembly to the Global Assembly Cache as part of the VSTO installation.

It's very important to test and debug your ECTs. They're run by the BCS runtime in the background and you can't debug ECTs from Visual Studio when they're running on client machines. Our recommendation is that you test and debug your ECTs on the server side by creating external lists in SharePoint. When you view and change the content of an external list on a SharePoint List page, you're running the ECT code associated with the external list and you can debug the ECT from a Visual Studio instance running on the same SharePoint server.

Finally, it's vitally important that you understand the BCS API entity operation mode. In the client-side application using BCS APIs to manipulate ECT entity instances, there are four operation modes you can use to manage ECT data. Here's a quick overview of how they work:

- **OperationMode.CacheWithoutRefresh:** When this operation mode is used for an entity instance, BCS returns the entity instance from the cache. If data isn't in the cache, BCS refreshes the cache from external systems and returns the cached copy. If the external systems can't be contacted, BCS throws an exception.
- **OperationMode.CacheWithImmedicateRefresh:** With this operation mode, BCS refreshes the entity instance in the cache first from the external system and then returns the cached copy. If the external system can't be contacted, BCS still returns the cached copy. If the entity instance is not cached and the external system can't be contacted, BCS throws an exception.
- **OperationMode.Offline:** With offline operation mode, external systems are never contacted even if there's no data in the cache. BCS returns the entity instance from the cache. If it's not there, BCS throws an exception.
- **OperationMode.Online:** For online operation mode, BCS will never use local cached data and will always contact external systems to get a copy of the entity instance. If the external system can't be contacted, BCS throws an exception.

Wrapping Up

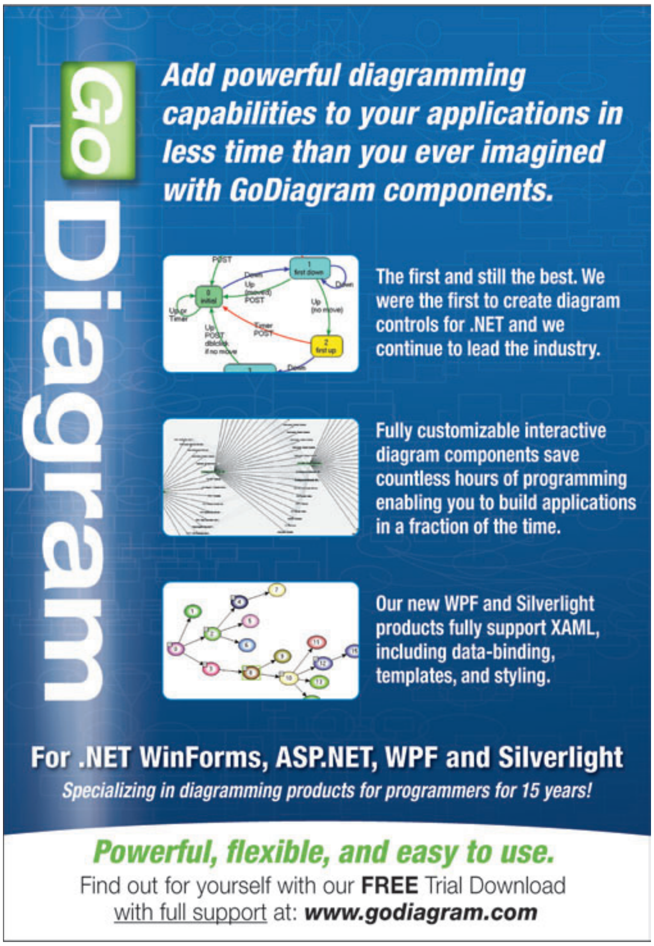
Microsoft Office and SharePoint 2010 with BCS provide the foundation for implementing a simple, easy-to-use and powerful solution for businesses to manage enterprise data from multiple external systems with familiar Office UIs. This foundation facilitates business data and process availability anywhere and anytime, therefore increasing business productivity.

It's important to test and debug your ECTs.

BCS synchronization infrastructure solves many problems associated with data copies, changes and conflicts. One of the advantages of BCS synchronization over other data-synchronization frameworks is that BCS allows your solutions to embed business and data-validation logic in the synchronization process through ECT implementations. Another benefit of BCS is the ability to synchronize composite data from multiple disparate systems (again through ECT design), instead of the point-to-point data synchronization that many other synchronization frameworks provide. ■

YING XIONG *is a principal architect on the Microsoft IT team, architecting and designing enterprise applications and integrations within Microsoft. He works on a variety of Microsoft technologies for building distributed enterprise applications. You can reach Xiong at yingx@microsoft.com.*

THANKS to the following technical experts for reviewing this article: *Rolando Jimenez Salgado and Satish Thatte*



It’s All About the SynchronizationContext

Stephen Cleary

Multithreaded programming can be quite difficult, and there’s a tremendous body of concepts and tools to learn when one embarks on this task. To help out, the Microsoft .NET Framework provides the SynchronizationContext class. Unfortunately, many developers aren’t even aware of this useful tool.

Regardless of the platform—whether it’s ASP.NET, Windows Forms, Windows Presentation Foundation (WPF), Silverlight or others—all .NET programs include the concept of SynchronizationContext, and all multithreading programmers can benefit from understanding and applying it.

This article discusses a prerelease version of Visual Studio Async. All information is subject to change.

This article discusses:

- The need for SynchronizationContext
- The concept of SynchronizationContext
- The implementations of SynchronizationContext
- AsyncOperationManager and AsyncOperation
- Examples of library support for SynchronizationContext

Technologies discussed:

ASP.NET, Windows Forms, Windows Presentation Foundation, Silverlight

The Need for SynchronizationContext

Multithreaded programs existed well before the advent of the .NET Framework. These programs often had the need for one thread to pass a unit of work to another thread. Windows programs were centered on message loops, so many programmers used this built-in queue to pass units of work around. Each multithreaded program that wanted to use the Windows message queue in this fashion had to define its own custom Windows message and convention for handling it.

When the .NET Framework was first released, this common pattern was standardized. At that time, the only GUI application type that .NET supported was Windows Forms. However, the framework designers anticipated other models, and they developed a generic solution. ISynchronizeInvoke was born.

The idea behind ISynchronizeInvoke is that a “source” thread can queue a delegate to a “target” thread, optionally waiting for that delegate to complete. ISynchronizeInvoke also provided a property to determine whether the current code was already running on the target thread; in this case, queuing up the delegate would be unnecessary. Windows Forms provided the only implementation of ISynchronizeInvoke, and a pattern was developed for designing asynchronous components, so everyone was happy.

Version 2.0 of the .NET Framework contained many sweeping changes. One of the major improvements was introducing asynchronous pages to the ASP.NET architecture. Prior to the .NET

Framework 2.0, every ASP.NET request needed a thread until the request was completed. This was an inefficient use of threads, because creating a Web page often depends on database queries and calls to Web services, and the thread handling that request would have to wait until each of those operations finished. With asynchronous pages, the thread handling the request could begin each of the operations and then return back to the ASP.NET thread pool; when the operations finished, another thread from the ASP.NET thread pool would complete the request.

However, ISynchronizeInvoke wasn't a good fit for the ASP.NET asynchronous pages architecture. Asynchronous components developed using the ISynchronizeInvoke pattern wouldn't work correctly within ASP.NET pages because ASP.NET asynchronous pages aren't associated with a single thread. Instead of queuing work to the original thread, asynchronous pages only need to maintain a *count* of outstanding operations to determine when the page request can be completed. After much thought and careful design, ISynchronizeInvoke was replaced by SynchronizationContext.

The Concept of SynchronizationContext

ISynchronizeInvoke satisfied two needs: determining if synchronization was necessary, and queuing a unit of work from one thread to another. SynchronizationContext was designed to replace ISynchronizeInvoke, but after the design process, it turned out to not be an exact replacement.

One aspect of SynchronizationContext is that it provides a way to queue a unit of work to a context. Note that this unit of work is queued to a *context* rather than a specific thread. This distinction is important, because many implementations of SynchronizationContext aren't based on a single, specific thread. SynchronizationContext does not include a mechanism to determine if synchronization is necessary, because this isn't always possible.

Another aspect of SynchronizationContext is that every thread has a "current" context. A thread's context isn't necessarily unique; its context instance may be shared with other threads. It's possible for a thread to change its current context, but this is quite rare.

A third aspect of SynchronizationContext is that it keeps a count of outstanding asynchronous operations. This enables the use of ASP.NET asynchronous pages and any other host needing this kind of count. In most cases, the count is incremented when the current SynchronizationContext is captured, and the count is decremented when the captured SynchronizationContext is used to queue a completion notification to the context.

There are other aspects of SynchronizationContext, but they're less important to most programmers. The most important aspects are illustrated in **Figure 1**.

The Implementations of SynchronizationContext

The actual "context" of the SynchronizationContext isn't clearly defined. Different frameworks and hosts are free to define their own context. Understanding these different implementations and their limitations clarifies exactly what the SynchronizationContext concept does and doesn't guarantee. I'll briefly discuss some of these implementations.

WindowsFormsSynchronizationContext (*System.Windows.Forms.dll: System.Windows.Forms*) Windows Forms apps will create and install a WindowsFormsSynchronizationContext as the current context for any thread that creates UI controls. This SynchronizationContext uses the ISynchronizeInvoke methods on a UI control, which passes the delegates to the underlying Win32 message loop. The context for WindowsFormsSynchronizationContext is a single UI thread.

Figure 1 **Aspects of the SynchronizationContext API**

```
// The important aspects of the SynchronizationContext APIclass
SynchronizationContext
{
  // Dispatch work to the context.
  void Post(..); // (asynchronously)
  void Send(..); // (synchronously)

  // Keep track of the number of asynchronous operations.
  void OperationStarted();
  void OperationCompleted();

  // Each thread has a current context.
  // If "Current" is null, then the thread's current context is
  // "new SynchronizationContext()", by convention.
  static SynchronizationContext Current { get; }
  static void SetSynchronizationContext(SynchronizationContext);
}
```

All delegates queued to the WindowsFormsSynchronizationContext are executed one at a time; they're executed by a specific UI thread in the order they were queued. The current implementation creates one WindowsFormsSynchronizationContext for each UI thread.

DispatcherSynchronizationContext (*WindowsBase.dll: System.Windows.Threading*) WPF and Silverlight applications use a DispatcherSynchronizationContext, which queues delegates to the UI thread's Dispatcher with "Normal" priority. This SynchronizationContext is installed as the current context when a thread begins its Dispatcher loop by calling Dispatcher.Run. The context for DispatcherSynchronizationContext is a single UI thread.

One aspect of SynchronizationContext is that it provides a way to queue a unit of work to a context.

All delegates queued to the DispatcherSynchronizationContext are executed one at a time by a specific UI thread in the order they were queued. The current implementation creates one DispatcherSynchronizationContext for each top-level window, even if they all share the same underlying Dispatcher.

Default (ThreadPool) SynchronizationContext (*mscorlib.dll: System.Threading*) The default SynchronizationContext is a default-constructed SynchronizationContext object. By convention, if a thread's current SynchronizationContext is null, then it implicitly has a default SynchronizationContext.

The default SynchronizationContext queues its asynchronous delegates to the ThreadPool but executes its synchronous delegates directly on the calling thread. Therefore, its context covers

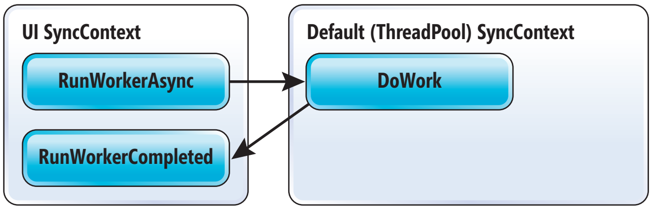


Figure 2 **A Single BackgroundWorker in a UI Context**

all ThreadPool threads as well as any thread that calls Send. The context "borrows" threads that call Send, bringing them into its context until the delegate completes. In this sense, the default context may include *any* thread in the process.

The default SynchronizationContext is applied to ThreadPool threads unless the code is hosted by ASP.NET. The default SynchronizationContext is also implicitly applied to explicit child threads (instances of the Thread class) unless the child thread sets its own SynchronizationContext. Thus, UI applications usually have two synchronization contexts: the UI SynchronizationContext covering the UI thread, and the default SynchronizationContext covering the ThreadPool threads.

Many event-based asynchronous components don't work as expected with the default SynchronizationContext.

Many event-based asynchronous components don't work as expected with the default SynchronizationContext. An infamous example is a UI application where one BackgroundWorker starts another BackgroundWorker. Each BackgroundWorker captures and uses the SynchronizationContext of the thread that calls RunWorkerAsync and later executes its RunWorkerCompleted event in that context. In the case of a single BackgroundWorker, this is usually a UI-based SynchronizationContext, so RunWorkerCompleted is executed in the UI context captured by RunWorkerAsync (see **Figure 2**).

However, if the BackgroundWorker starts another BackgroundWorker from its DoWork handler, then the nested BackgroundWorker doesn't capture the UI SynchronizationContext. DoWork is executed by a ThreadPool thread with the default SynchronizationContext. In this case, the nested RunWorkerAsync will capture the default SynchronizationContext, so it will execute its RunWorkerCompleted on a ThreadPool thread instead of a UI thread (see **Figure 3**).

By default, all threads in console applications and Windows Services only have the default SynchronizationContext. This causes some event-based asynchronous components to fail. One solution for this is to create an explicit child thread and install a SynchronizationContext on that thread, which can then provide a context for these components. Implementing a SynchronizationContext is beyond the scope of this article, but the ActionThread class of the Nito.Async library (nitoasync.codeplex.com) may be used as a general-purpose SynchronizationContext implementation.

AspNetSynchronizationContext (*System.Web.dll: System.Web [internal class]*) The ASP.NET SynchronizationContext is installed on thread pool threads as they execute page code. When a delegate is queued to a captured AspNetSynchronizationContext, it restores the identity and culture of the original page and then executes the delegate directly. The delegate is directly invoked even if it's "asynchronously" queued by calling Post.

Conceptually, the context of AspNetSynchronizationContext is complex. During the lifetime of an asynchronous page, the context starts with just one thread from the ASP.NET thread pool. After the asynchronous requests have started, the context doesn't include any threads. As the asynchronous requests complete, the thread pool threads executing their completion routines enter the context. These may be the same threads that initiated the requests but more likely would be whatever threads happen to be free at the time the operations complete.

If multiple operations complete at once for the same application, AspNetSynchronizationContext will ensure that they execute one at a time. They may execute on any thread, but that thread will have the identity and culture of the original page.

One common example is a WebClient used from within an asynchronous Web page. DownloadDataAsync will capture the current SynchronizationContext and later will execute its DownloadDataCompleted event in that context. When the page begins executing, ASP.NET will allocate one of its threads to execute the code in that page. The page may invoke DownloadDataAsync and then return; ASP.NET keeps a count of the outstanding asynchronous operations, so it understands that the page isn't complete. When the WebClient object has downloaded the requested data, it will receive notification on a thread pool thread. This thread will raise DownloadDataCompleted in the captured context. The context will stay on the same thread but will ensure the event handler runs with the correct identity and culture.

Notes on SynchronizationContext Implementations

SynchronizationContext provides a means for writing components that may work within many different frameworks. BackgroundWorker and WebClient are two examples that are equally at home in Windows Forms, WPF, Silverlight, console and ASP.NET apps. However, there are some points that must be kept in mind when designing such reusable components.

Generally speaking, SynchronizationContext implementations aren't equality-comparable. This means that there's no equivalent to

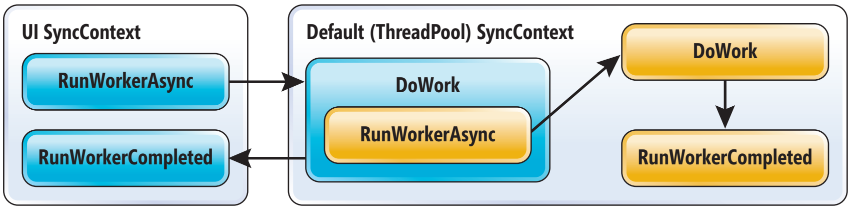


Figure 3 **Nested BackgroundWorkers in a UI Context**

ISynchronizeInvoke.InvokeRequired. However, this isn't a tremendous drawback; code is cleaner and easier to verify if it always executes within a known context instead of attempting to handle multiple contexts.

Not all SynchronizationContext implementations guarantee the order of delegate execution or synchronization of delegates. The UI-based SynchronizationContext implementations do satisfy these conditions, but the ASP.NET SynchronizationContext only provides synchronization. The default SynchronizationContext doesn't guarantee either order of execution or synchronization.

There isn't a 1:1 correspondence between SynchronizationContext instances and threads. The WindowsFormsSynchronizationContext does have a 1:1 mapping to a thread (as long as SynchronizationContext.CreateCopy isn't invoked), but this isn't true of any of the other implementations. In general, it's best to not assume that any context instance will run on any specific thread.

Finally, the SynchronizationContext.Post method isn't necessarily asynchronous. Most implementations do implement it asynchronously, but AspNetSynchronizationContext is a notable exception. This may cause unexpected re-entrancy issues. A summary of these different implementations can be seen in **Figure 4**.

AsyncOperationManager and AsyncOperation

The AsyncOperationManager and AsyncOperation classes in the .NET Framework are lightweight wrappers around the SynchronizationContext abstraction. AsyncOperationManager captures the current SynchronizationContext the first time it creates an AsyncOperation, substituting a default SynchronizationContext if the current one is null. AsyncOperation posts delegates asynchronously to the captured SynchronizationContext.

Most event-based asynchronous components use AsyncOperationManager and AsyncOperation in their implementation. These work well for asynchronous operations that have a defined point of completion—that is, the asynchronous operation begins at one point and ends with an event at another. Other asynchronous notifications may not have a defined point of completion; these may be a type of subscription, which begins at one point and then continues indefinitely. For these types of operations, SynchronizationContext may be captured and used directly.

New components shouldn't use the event-based asynchronous pattern. The Visual Studio asynchronous Community Technology Preview (CTP) includes a document describing the task-based asynchronous pattern, in which components return Task and Task<TResult> objects instead of raising events through SynchronizationContext. Task-based APIs are the future of asynchronous programming in .NET.

Examples of Library Support for SynchronizationContext

Simple components such as BackgroundWorker and WebClient are implicitly portable by themselves, hiding the SynchronizationContext capture and usage. Many libraries have a more visible use of SynchronizationContext. By exposing APIs using SynchronizationContext, libraries not only gain framework independence, they also provide an extensibility point for advanced end users.

In addition to the libraries I'll discuss now, the current SynchronizationContext is considered to be part of the ExecutionContext. Any system that captures a thread's ExecutionContext captures the current SynchronizationContext. When the ExecutionContext is restored, the SynchronizationContext is usually restored as well.

Windows Communication Foundation (WCF): UseSynchronizationContext WCF has two attributes that are used to configure server and client behavior: ServiceBehaviorAttribute and CallbackBehaviorAttribute. Both of these attributes have a Boolean property: UseSynchronizationContext. The default value of this attribute is true, which means that the current SynchronizationContext is captured when the communication channel is created, and this captured SynchronizationContext is used to queue the contract methods.

Task-based APIs are the future of asynchronous programming in .NET.

Normally, this behavior is exactly what is needed: Servers use the default SynchronizationContext, and client callbacks use the appropriate UI SynchronizationContext. However, this can cause problems when re-entrancy is desired, such as a client invoking a server method that invokes a client callback. In this and similar cases, the WCF automatic usage of SynchronizationContext may be disabled by setting UseSynchronizationContext to false.

This is just a brief description of how WCF uses SynchronizationContext. See the article "Synchronization Contexts in WCF" (msdn.microsoft.com/magazine/cc163321) in the November 2007 issue of *MSDN Magazine* for more details.

Windows Workflow Foundation (WF): WorkflowInstance.SynchronizationContext WF hosts originally used WorkflowSchedulerService and derived types to control how workflow

Figure 5 **Progress Reporting with UI Updates**

```
private void button1_Click(object sender, EventArgs e)
{
  // This TaskScheduler captures SynchronizationContext.Current.
  TaskScheduler taskScheduler = TaskScheduler.FromCurrentSynchronizationContext();

  // Start a new task (this uses the default TaskScheduler,
  // so it will run on a ThreadPool thread).
  Task.Factory.StartNew(() =>
  {
    // We are running on a ThreadPool thread here.

    ; // Do some work.

    // Report progress to the UI.
    Task reportProgressTask = Task.Factory.StartNew(() =>
      {
        // We are running on the UI thread here.

        ; // Update the UI with our progress.
      },
      CancellationToken.None,
      TaskCreationOptions.None,
      taskScheduler);
    reportProgressTask.Wait();

    ; // Do more work.
  });
}
```

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

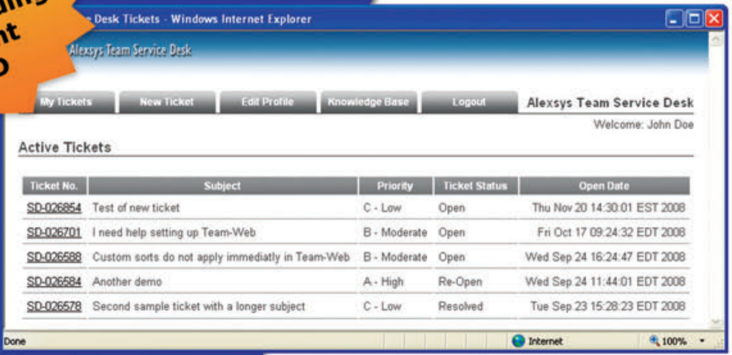
Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD

Alexsys Team


Desk Tickets - Windows Internet Explorer
Alexsys Team Service Desk
My Tickets
New Ticket
Edit Profile
Knowledge Base
Logout
Alexsys Team Service Desk
Done
Internet
100%

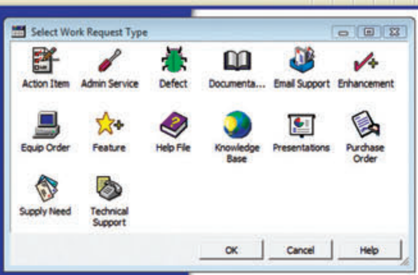
Welcome: John Doe

Active Tickets

Ticket No.	Subject	Priority	Ticket Status	Open Date
SD-026854	Test of new ticket	C - Low	Open	Thu Nov 20 14:30:01 EST 2008
SD-026701	I need help setting up Team-Web	B - Moderate	Open	Fri Oct 17 09:24:32 EDT 2008
SD-026588	Custom sorts do not apply immediatly in Team-Web	B - Moderate	Open	Wed Sep 24 16:24:47 EDT 2008
SD-026584	Another demo	A - High	Re-Open	Wed Sep 24 11:44:01 EDT 2008
SD-026578	Second sample ticket with a longer subject	C - Low	Resolved	Tue Sep 23 15:28:23 EDT 2008

New in Team 2.11

- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment



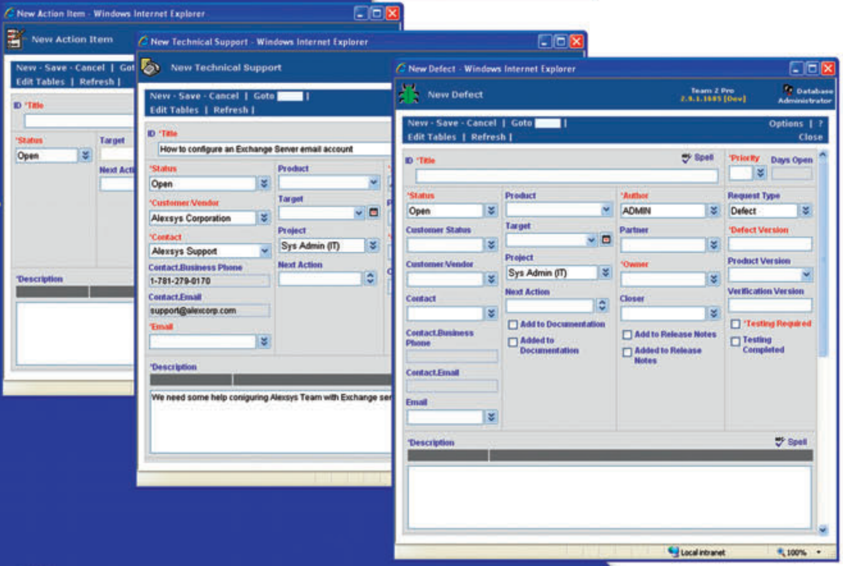
Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration

ZDNet Editors' Pick



New Action Item: *Status: Open

New Technical Support:
ID *Title: How to configure an Exchange Server email account
*Status: Open
Product:
*Customer/Vendor: Alexsys Corporation
Target:
*Contact: Alexsys Support
Project: Sys Admin (IT)
Contact.Business Phone: 1-781-279-0170
Next Action:
Contact.Email: support@alexcorp.com
*Email:
*Description: We need some help coniguring Alexsys Team with Exchange ser

New Defect:
ID *Title:
*Priority:
Days Open:
*Status: Open
Product:
*Author: ADMIN
Request Type: Defect
Customer Status:
Target:
Partner:
*Defect Version:
Customer/Vendor:
Project: Sys Admin (IT)
*Owner:
Product Version:
Contact:
Next Action:
Closer:
Verification Version:
□ Add to Documentation
□ Added to Documentation
□ Add to Release Notes
□ Added to Release Notes
□ *Testing Required
□ Testing Completed
Contact.Business Phone:
Contact.Email:
Email:
*Description:

Free Trial and Single User FreePack™ available at www.alexcorp.com.
FreePack™ includes a free single user Team Pro and Team-Web license.
Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.

Figure 4 **Summary of SynchronizationContext Implementations**

	Specific Thread Used to Execute Delegates	Exclusive (Delegates Execute One at a Time)	Ordered (Delegates Execute in Queue Order)	Send May Invoke Delegate Directly	Post May Invoke Delegate Directly
Windows Forms	Yes	Yes	Yes	If called from UI thread	Never
WPF/Silverlight	Yes	Yes	Yes	If called from UI thread	Never
Default	No	No	No	Always	Never
ASP.NET	No	Yes	No	Always	Always

activities were scheduled on threads. Part of the .NET Framework 4 upgrade included the SynchronizationContext property on the WorkflowInstance class and its derived WorkflowApplication class.

The SynchronizationContext may be set directly if the hosting process creates its own WorkflowInstance. SynchronizationContext is also used by WorkflowInvoker.InvokeAsync, which captures the current SynchronizationContext and passes it to its internal WorkflowApplication. This SynchronizationContext is then used to post the workflow completion event as well as the workflow activities.

Task Parallel Library (TPL): TaskScheduler.FromCurrentSynchronizationContext and CancellationToken.Register The TPL uses task objects as its units of work and executes them via a TaskScheduler. The default TaskScheduler acts like the default SynchronizationContext, queuing the tasks to the ThreadPool. There's another TaskScheduler provided by the TPL queue that queues tasks to a SynchronizationContext. Progress reporting with UI updates may be done with a nested task, as shown in **Figure 5**.

There isn't a 1:1 correspondence between SynchronizationContext instances and threads.

The CancellationToken class is used for any type of cancellation in the .NET Framework 4. To integrate with existing forms of cancellation, this class allows registering a delegate to invoke when cancellation is requested. When the delegate is registered, a SynchronizationContext may be passed. When the cancellation is requested, CancellationToken queues the delegate to the SynchronizationContext instead of executing it directly.

Microsoft Reactive Extensions (Rx): ObserveOn, SubscribeOn and SynchronizationContextScheduler Rx is a library that treats events as streams of data. The ObserveOn operator queues events through a SynchronizationContext, and the SubscribeOn operator queues the *subscriptions* to those events through a SynchronizationContext. ObserveOn is commonly used to update the UI with incoming events, and SubscribeOn is used to consume events from UI objects.

Rx also has its own way of queuing units of work: the IScheduler interface. Rx includes SynchronizationContextScheduler, an implementation of IScheduler that queues to a SynchronizationContext.

Visual Studio Async CTP: await, ConfigureAwait, SwitchTo and EventProgress<T> The Visual Studio support for asynchronous code transformations was announced at the Microsoft Professional Developers Conference 2010. By default, the current SynchronizationContext is captured at an await point, and this SynchronizationContext is used to resume after the await (more precisely, it captures the current SynchronizationContext *unless it is null*, in which case it captures the current TaskScheduler):

```
private async void button1_Click(object sender, EventArgs e)
{
  // SynchronizationContext.Current is implicitly captured by await.
  var data = await webClient.DownloadStringTaskAsync(uri);

  // At this point, the captured SynchronizationContext was used to resume
  // execution, so we can freely update UI objects.
}
```

ConfigureAwait provides a means to avoid the default SynchronizationContext capturing behavior; passing false for the flowContext parameter prevents the SynchronizationContext from being used to resume execution after the await. There's also an extension method on SynchronizationContext instances called SwitchTo; this allows any async method to change to a different SynchronizationContext by invoking SwitchTo and awaiting the result.

The asynchronous CTP introduces a common pattern for reporting progress from asynchronous operations: the IProgress<T> interface and its implementation EventProgress<T>. This class captures the current SynchronizationContext when it's constructed and raises its ProgressChanged event in that context.

In addition to this support, void-returning async methods will increment the asynchronous operation count at their start and decrement it at their end. This behavior makes void-returning async methods act like top-level asynchronous operations.

Limitations and Guarantees

Understanding SynchronizationContext is helpful for any programmer. Existing cross-framework components use it to synchronize their events. Libraries may expose it to allow advanced flexibility. The savvy coder who understands the limitations and guarantees of SynchronizationContext is better able to write and consume such classes. ■

STEPHEN CLEARY *has had an interest in multithreading ever since he first heard of the concept. He's completed many business-critical multitasking systems for major clients including Syracuse News, R. R. Donnelley and BlueScope Steel. He regularly speaks at .NET user groups, BarCamps and Day of .NET events near his home in Northern Michigan, usually on a multithreading topic. He maintains a programming blog at nitoprograms.com.*

THANKS to the following technical expert for reviewing this article:
Eric Eilebrecht

Visual Studio TFS Branching and Merging Guidance

Bill Heys and Willy-Peter Schaub

Since its inception in 2006, the Visual Studio ALM Rangers team has operated within the Microsoft developer division to promote collaboration between the Visual Studio product groups, Microsoft Services and the Microsoft Most Valuable Professional (MVP) community. The standard vision statement of the Rangers team is to *"accelerate the adoption of Visual Studio with out-of-band solutions for missing features or guidance"* by addressing missing functionality and removing adoption blockers. The collaboration among the variety of technology and business experts allows the Rangers to empower communities by sharing real-world experience. (You can learn more about the Rangers at msdn.microsoft.com/vstudio/ee358786.)

The Visual Studio Team Foundation Server (TFS) Branching Guide 2010 (tfsbranchingguideiii.codeplex.com) consolidates insightful and practical guidance around branching and merging with Visual Studio TFS 2010 by providing hands-on labs and lessons learned from the community. In this article, we'll introduce you to some of the advanced branching scenarios that we're working on for the next guidance version.

This article discusses:

- The history of branching guidance
- Guidance goals for the future
- The Feature Team branching scenario
- The common code-sharing branching scenario
- The architecture tooling and modeling branching scenario

Technologies discussed:

Visual Studio Team Foundation Server 2010

Code download available at:

code.msdn.microsoft.com/mag201102ALM

Branching: 'Today's State of the Nation'

The Rangers Branching Guidance started as a Rangers project after Visual Studio 2005 and TFS 2005 were released. This first release of the Rangers guidance was published on CodePlex in 2007.

In 2008, the Rangers kicked off the Branching Guidance II Project. For this second release, we reorganized the guidance into a set of related documents (Main, Scenarios, Q&A, Diagrams, Posters and so on). Each of the secondary documents was intended to build upon the primary guidance as presented in the Main branching document. Ranger Branching Guidance II was published on CodePlex in late 2008.

In 2009, the Rangers team once again kicked off a new Branching Guidance project: Branching Guidance 2010. This third release focused on showing many new branching features in Visual Studio 2010 and TFS 2010. A key new feature in 2010 is branch visualization.

In part because the latest version is titled Rangers Visual Studio TFS Branching Guide 2010, there has been some apparent confusion whether this guidance applies exclusively to Visual Studio 2010. We want to make it clear that the best practices and guidance presented in the 2010 guidance documents can continue to be applied to earlier versions of Visual Studio and TFS. In fact, the

Rangers team has received positive feedback from people using other tools for Source Control Management (SCM).

For 2011, the Rangers team is once again planning an update to the Rangers Branching Guidance.

Feel free to post questions, candid feedback or concerns to the CodePlex site.

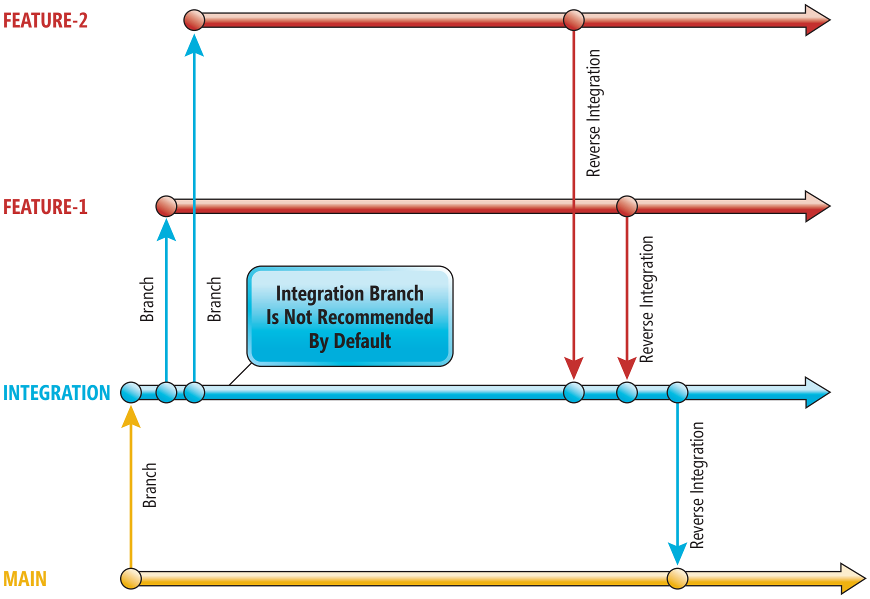


Figure 1 **Main and Integration Branches**

Branching Goals and Strategies

A key goal of branching is to provide isolation between parallel streams of work. In the current Rangers Branching Guidance 2010, we tended to focus more on release isolation than on isolation during a complex development initiative.

In many cases, all development activities for the next release of a product can be owned by a single development team. In this simple case, only one development branch is needed to isolate development from ongoing stabilization (Main branch) or sustained engineering (shipping production releases, along with ongoing hotfix and service pack support).

The Rangers often get asked about support for more complex development initiatives where a single development branch doesn't provide enough flexibility or isolation for a larger product development effort. In the next release of the Rangers Branching Guidance, we'll be adding more direction in addressing complex development scenarios such as feature team development.

We like to separate branching strategy discussions into two areas:

1. How does my organization *develop* software? Do we have a smaller, simpler team structure or do we need to support more complex teams with parallel development efforts?
2. How does my organization *release* software to its customers, either internal or external? Do we need to support multiple released versions? Do we need to provide hotfixes or service packs?

In some scenarios, an organization's release strategy may influence the development process, particularly the structure of the development teams. Often, however, the complexity of the release process and branching strategy can be independent from the complexity of the development process and branching strategy.

In designing a branching strategy, consider not only the branching structure, but also the branching process. For example, in the basic branch plan described in Rangers Branching Guidance 2010, there are only three branches (Main, Development and Release). A good branching strategy will describe the branch relationships (for example, Main is a parent to both the Development branch and the Release branch).

In addition, a branching strategy should describe the process implied by the branching structure. For example, how often do you build code in the Main branch? How often do you merge code (forward integration) from the Main branch to the Development branches? What are the conditions for merging code (reverse integration) from a Development branch back to the Main branch and so on? Let's discuss some typical branching scenarios.

The Feature Team Scenario

Organizations often need a branching strategy to support large, complex development efforts involving multiple development teams or feature teams working in parallel. Questions arise about how many separate development branches are needed. If I have multiple development branches, when and how do I integrate features developed by one team with features developed by other teams? Answers to these questions should be incorporated into a development-branching strategy.

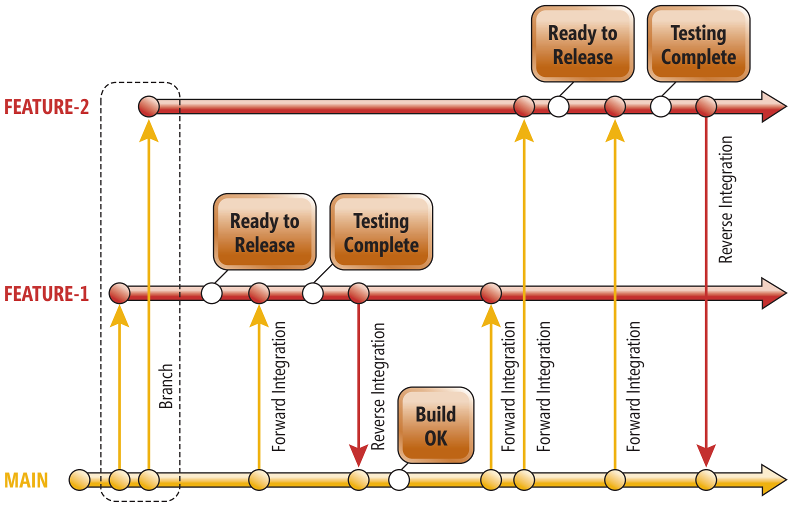
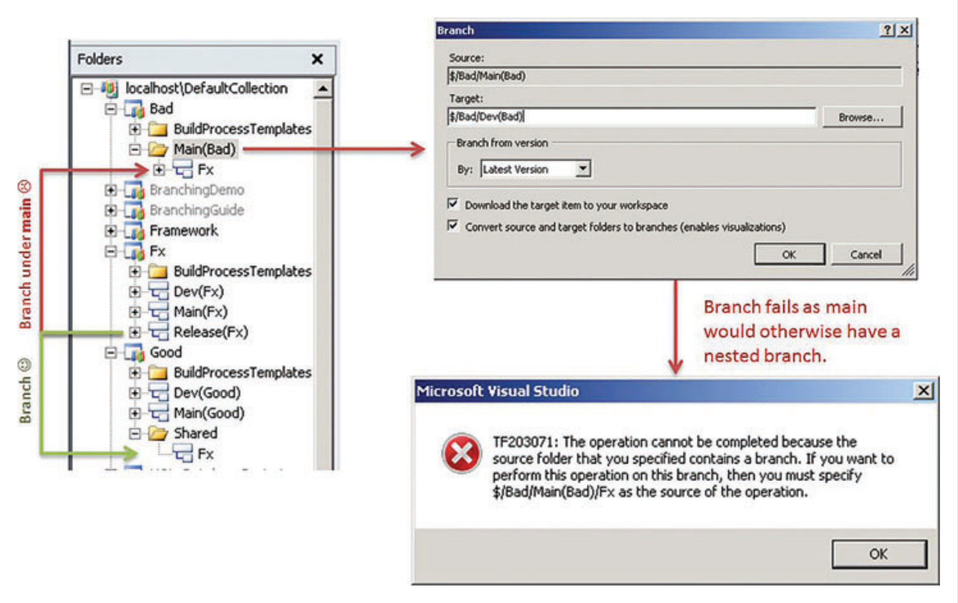


Figure 2 **Feature Branching**



Source: $/Bad/Main(Bad)

Target: $/Bad/Dev(Bad)

Branch from version

By: Latest Version

☑ Download the target item to your workspace

☑ Convert source and target folders to branches (enables visualizations)

Branch fails as main would otherwise have a nested branch.

TF203071: The operation cannot be completed because the source folder that you specified contains a branch. If you want to perform this operation on this branch, then you must specify $/Bad/Main(Bad)/Fx as the source of the operation.

Figure 3 **Example of a Nested Branch Causing an Error in Team Foundation Server 2010**

Let's start by describing a complex development initiative. Although there may be a common release schedule for the entire initiative, there may be several separate feature teams working on independent milestones. As these features are completed and tested, they'll be integrated into the Main branch.

On a single team, individual developers use local workspaces to isolate their changes from others on their team. Feature Team branches are a good way to isolate changes made by one Feature Team from changes made by other Feature Teams working in parallel on the same product. Without Feature Team isolation, changes made by one team may introduce breaking changes that impact the velocity of other teams.

Creating the branching structure for Feature Team isolation is relatively straightforward. But first, we need to plan for how the Feature Team branches will be integrated later on. Do we add an "integration branch" in between the Main branch and the Feature Team branches, as shown in **Figure 1**?

Or do we eliminate the integration layer and integrate the Feature Team changes another way? What's the best practice recommendation?

We recommend minimizing the number of levels in a branching hierarchy. Adding an integration layer between Main and the Feature Team branches effectively doubles the merges required to move changes between the Main branch and the Feature branches. Branching helps isolate changes, but the cost of branching is the resulting effort needed to merge code between branches and to resolve merge conflicts that always seem to present themselves. Adding an integration layer doubles the merges and likely doubles the effort to resolve merge conflicts.

If we eliminate the integration layer, we reduce the number of layers in our branch hierarchy. But where does the integration of Feature Team 1 and Feature Team 2 happen, and where is the integration tested? In order to keep the Main branch as stable as possible, avoid introducing untested integration changes into the Main branch. Without an integration layer, the merging of features and testing the integration must be done in a controlled fashion in the Feature Team branches themselves.

We recommend doing daily builds in the Main (stable) branch and, following a good daily build, doing a merge from Main to the Development (Feature) branches. Don't merge code from a Feature branch back to Main until the code in the Feature branch is relatively stable. In other words, the Feature should pass quality assurance gates before it's merged with Main.

Only when the code in a Feature branch is deemed "ready to release" or "ready to share with other teams" should we consider integrating this Feature branch with Main or with other Feature branches. **Figure 2** illustrates this process after each "ready to release" milestone.

Following are the process steps:

- Before merging Feature Team 1 branch with Main, do one final merge (forward integration, or FI) from Main to the Feature Team 1 branch.
- Complete a final test of this integration of code from Main with code in the Feature Team 1 branch.
- Once the code in the Feature Team 1 branch is stable, merge this code (reverse integration, or RI) back to Main.
- At this point, the code in Main incorporates the code from Feature Team 1.
- Perform a build and test in Main, equivalent to the daily build. On the next successful build of Main, merge Main to each of the Feature Team branches. Initially, this will result in Feature Team 1 code being merged with the code in Feature Team 2.
- Test the integration of Feature Team 1 code with Feature Team 2 code in the Feature Team 2 branch.

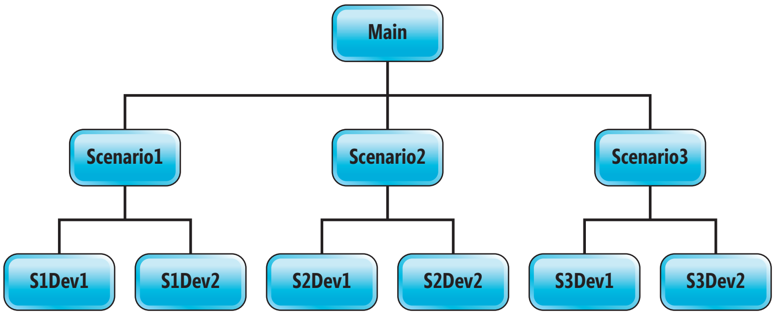
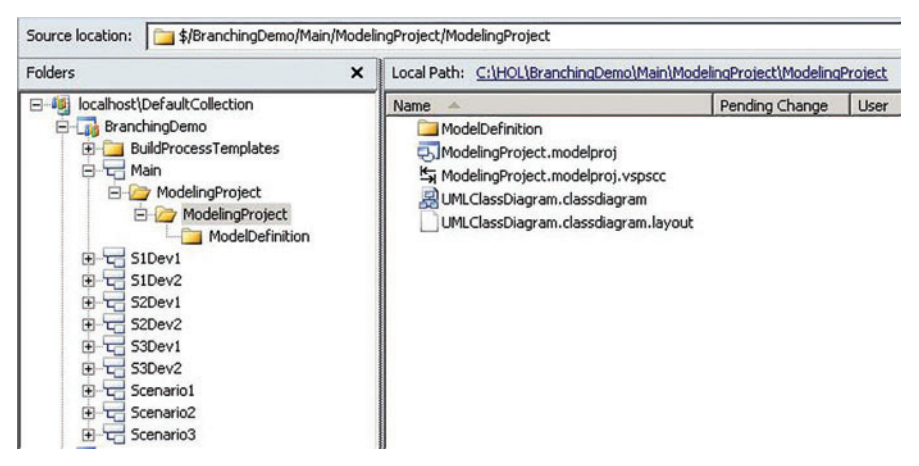


Figure 4 **Evaluation Scenarios in a Test Environment to Test Possibility of Branching and Merging**

- When Feature Team 2 code is ready for release or ready to share with other teams, merge Feature Team 2 code back to Main. But first do one final merge from Main to Feature Team 2 and test the final integration.

Note: A key requirement for omitting a separate integration layer is the ability to use automated testing for the integrations. Automated testing helps reduce the impact on code velocity (that is, feature team productivity) as the team works to identify and resolve bugs that arise from merging many changes into a branch.

If automated testing isn't available for testing integration changes, the risk is that code velocity of the feature teams will be adversely affected as they undertake manual testing to identify and resolve bugs. In this scenario, an organization might consider adding an integration layer between Main and the Feature branches. As we previously noted, the integration layer may result in increased merging and merge conflict resolution. But the benefit could be that having this layer may allow for integration with less impact on code velocity of the feature teams.


Source location: $/BranchingDemo/Main/ModelingProject/ModelingProject

Folders

```
localhost\DefaultCollection
  BranchingDemo
    BuildProcessTemplates
    Main
      ModelingProject
        ModelingProject
          ModelDefinition
    S1Dev1
    S1Dev2
    S2Dev1
    S2Dev2
    S3Dev1
    S3Dev2
    Scenario1
    Scenario2
    Scenario3
```

Local Path: C:\HOL\BranchingDemo\Main\ModelingProject\ModelingProject

Name	Pending Change	User
ModelDefinition		
ModelingProject.modelproj		
ModelingProject.modelproj.vspscc		
UMLClassDiagram.classdiagram		
UMLClassDiagram.classdiagram.layout		

Figure 5 **BranchingDemo Team Project, as Viewed in the Source Explorer**

A good branching strategy requires a sound branching structure coupled with a sound branching process to ensure maximum code velocity for the Feature Teams while at the same time maintaining the stability of the Main branch.

Common Code-Sharing Scenario

Sharing common code between projects is a challenge for many organizations. There are three main techniques for sharing code between projects or solutions in Visual Studio:

- File linking
- Binary (Assembly) sharing
- Source code sharing

As we discuss elsewhere in this article, there are also several techniques for code isolation:

- Team project isolation
- Branch isolation
- Workspace isolation

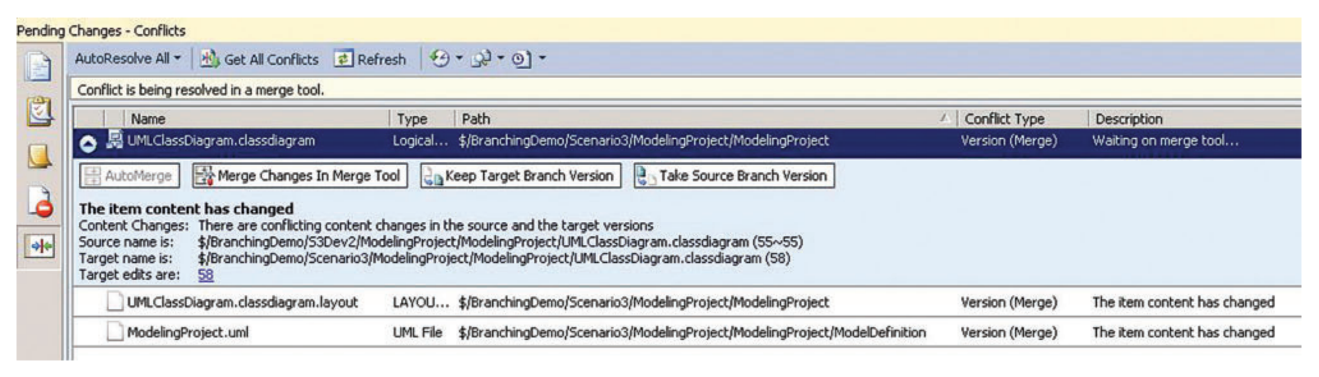
Choosing the correct code-sharing strategy for your organization likely involves a combination of code-sharing techniques and isolation techniques.

File Linking: This is a feature of Visual Studio (Add Existing Item) where multiple projects can share a reference to a single source file. File linking is better suited for small projects with a limited number of files being shared. (This resembles file sharing in Visual Source Safe.)

With file linking, there's only one version of the linked source file to maintain. Changes made to the linked file are immediately received by all projects linking to the file. The disadvantage of file linking is that changes to the linked file should be coordinated with all dependent project teams. Even carefully coordinated changes might cause breaking changes in dependent projects.

Binary Sharing (Assembly References): With binary sharing, a Visual Studio solution references common shared code using assembly references. Here, building or compiling the dependent solution doesn't also compile the common shared source code. Compiling dependent projects will be faster using assembly references rather than project references.

Teams that own the common code have full ownership and control, which in theory means that the control, versioning and quality of the product is probably better, and branching and merging complexities are avoided.


Pending Changes - Conflicts

AutoResolve All
Get All Conflicts
Refresh

Conflict is being resolved in a merge tool.

Name	Type	Path	Conflict Type	Description
UMLClassDiagram.classdiagram	Logical...	$/BranchingDemo/Scenario3/ModelingProject/ModelingProject	Version (Merge)	Waiting on merge tool...

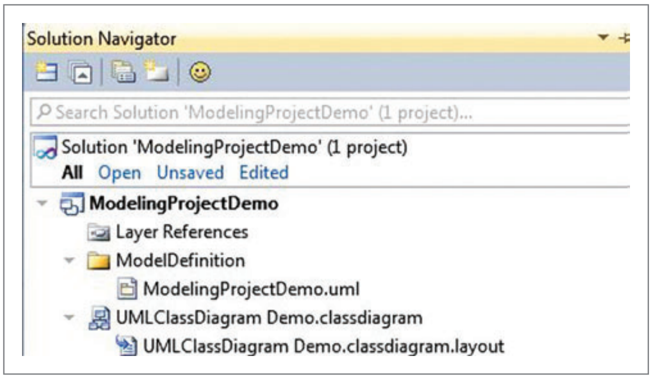
AutoMerge
Merge Changes In Merge Tool
Keep Target Branch Version
Take Source Branch Version

The item content has changed

```
Content Changes:  There are conflicting content changes in the source and the target versions
Source name is:   $/BranchingDemo/S3Dev2/ModelingProject/ModelingProject/UMLClassDiagram.classdiagram (55~55)
Target name is:   $/BranchingDemo/Scenario3/ModelingProject/ModelingProject/UMLClassDiagram.classdiagram (58)
Target edits are: 58
```

Name	Type	Path	Conflict Type	Description
UMLClassDiagram.classdiagram.layout	LAYOU...	$/BranchingDemo/Scenario3/ModelingProject/ModelingProject	Version (Merge)	The item content has changed
ModelingProject.uml	UML File	$/BranchingDemo/Scenario3/ModelingProject/ModelingProject/ModelDefinition	Version (Merge)	The item content has changed

Figure 6 **A Merge Causes Conflicts Due to the Changes Made by Two Teams of the Class1 Class**



Solution Navigator

Search Solution 'ModelingProjectDemo' (1 project)...

Solution 'ModelingProjectDemo' (1 project)

All Open Unsaved Edited

ModelingProjectDemo

Layer References

ModelDefinition

ModelingProjectDemo.uml

UMLClassDiagram Demo.classdiagram

UMLClassDiagram Demo.classdiagram.layout

Figure 7 **Visualizations Spread Across Three Main Files**

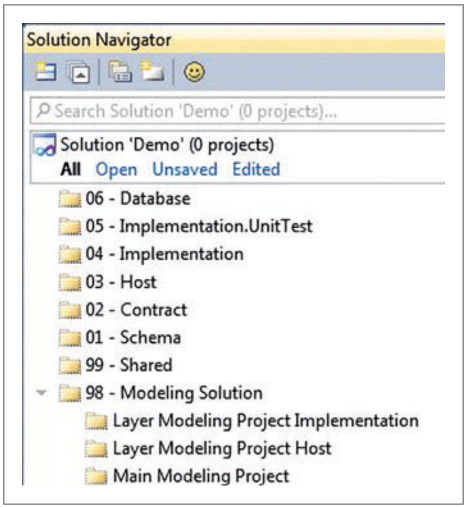
Because teams reusing the common code don't have access to the common source code, they're dependent on the owning team to add new features and resolve bugs in the common shared code.

The assemblies for the common code can be shared by copying them to a well-known file share that can be referenced by dependent projects. Signed assemblies may need to be added to the Global Assembly Cache. Alternatively, the assemblies can be copied from the common code Team Project to a bin folder under the dependent project's Main branch.

Source Code Sharing: With source code sharing in Visual Studio, a dependent project uses a project reference for the common shared code. When the solution is built, all projects are built, including the common shared code projects. With complex projects, having many project references to shared code may significantly increase build time.

In this scenario, the common shared code is owned and managed by a team in its own TFS Team Project. To share this common code, first branch the code to a folder with the consuming (dependent) project's Team Project like this:

- Create a folder within the dependent project's Team Project called "Share" (for example, $\Product1\Share).
- Branch the Main branch of the Common Library (for example, EnterpriseLibrary) to the Share folder of the dependent Project, for example branch $\Enterprise Library\Main to $\Product1\Share\EnterpriseLibrary.
- Add the appropriate common code projects to the dependent project's solution.
- Create project references from the dependent project to the existing common code projects in the solution.



Solution Navigator

Search Solution 'Demo' (0 projects)...

Solution 'Demo' (0 projects)

All Open Unsaved Edited

06 - Database

05 - Implementation.UnitTest

04 - Implementation

03 - Host

02 - Contract

01 - Schema

99 - Shared

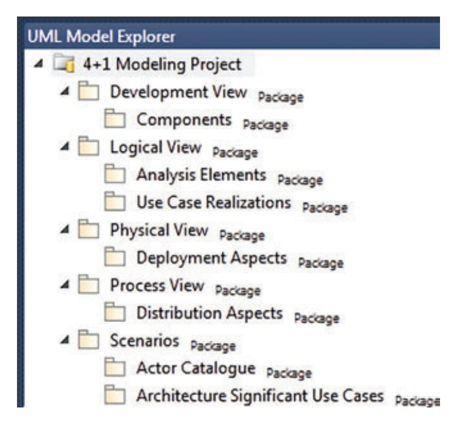
98 - Modeling Solution

Layer Modeling Project Implementation

Layer Modeling Project Host

Main Modeling Project

Figure 8 **The Proposed Package-Based Structure as Seen in Solution View**



UML Model Explorer

4+1 Modeling Project

Development View Package

Components Package

Logical View Package

Analysis Elements Package

Use Case Realizations Package

Physical View Package

Deployment Aspects Package

Process View Package

Distribution Aspects Package

Scenarios Package

Actor Catalogue Package

Architecture Significant Use Cases Package

Figure 9 **The Proposed Package-Based Structure as Seen in UML Model Explorer**

Note: Nested branches aren't supported in TFS 2010. A nested branch error might arise when you try to perform a branch operation that would cause a new branch to be created (either above or below) an existing branch in the folder structure (see **Figure 3***).*

Your organization needs to decide whether changes to the common shared source code should be allowed within each dependent project. To prevent changes, after branching from the Common Library the new branch can be made read-only. All changes to the common code source must then be made in the Common Library Team Project and merged into the dependent projects' Team Projects.

Alternatively, changes can be made to the shared code source within a dependent Team Project. These changes can be merged (through reverse integration) back to the Common Library Team Project. Your organization needs to carefully manage these changes to avoid incompatibilities that make merging these changes back to the Common Library difficult or impossible, perhaps resulting in multiple copies of the shared code.

Architecture Tooling and Modeling Scenario

In Visual Studio Ultimate, you can create UML and Layer models that exist in their own separate Visual Studio projects and can contain many packages, dealing with different parts of the solution (see the Architecture Tooling Guidance at vsarchitectureguide.codeplex.com and Modeling the Application at msdn.microsoft.com/library/57b85fsc.aspx).

To explore whether branching and merging is possible with models, we can create a simple test environment with three scenarios, as shown in **Figure 4**.

We can create a Main branch, with a solution that contains a model project with an empty UML class diagram as a hypothetical stable project. We can then branch Main to Scenario1, Scenario2 and Scenario3, then branch each Scenario to a Dev1 and Dev2 branch representing development teams, as shown in **Figure 5**.

It's obvious that we have no issue with branching, but can we *reverse integrate (merge) changes in the model*?

In Scenario1, the teams make no model changes, and with Scenerio2 only one of the two teams expands the models. The resultant RI from the Development branches to the associated scenario branch is uneventful with the unchanged Scenario1 model and the updated Scenario2 model.

Scenario3 is a more realistic example where both teams update the model. Dev1 Team creates two classes and Dev2 Team creates one class.

The assiduous reader will notice that both teams created a Class1 class, with different operations.

Reverse integrating the first of the two development branches back to the Scenario3 branch gives the false sense of security that the merge will be easy. But when the second team merges changes to the Scenario3 branch, conflicts for three files (.classdiagram, .layout and .uml) block the check-in, as shown in **Figure 6**.

We could select the options "Keep Target Branch Version" or "Take Source Branch Version" and answer the question whether merging is possible with a "yes." The result, however, would be a very unhappy team losing its changes. The alternative is to select the "Merge Changes in Merge Tool" option for a manual merge, which is impractical, unintuitive and error-prone for most of us.

The branching and merging of architecture models is therefore possible, but is it recommended? The problem with the UML model is that the visualizations—for example, the class diagram—are spread across three main files (.layout, .classdiagram and .uml) as shown in **Figure 7**.

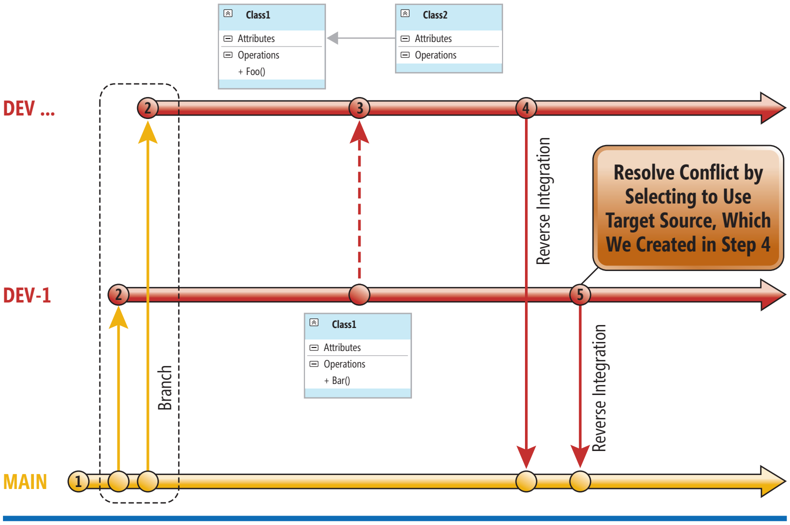


Figure 10 **A Manual Model Edit Merge**

The .layout file defines the size and positions of the shapes in the model. The .uml file is the "master" model, and the .classdiagram file holds a cache of the content from the .uml file, which is present in the diagram.

Merges are also difficult, as normal edits in the modeling tools are validated and often augmented by the tool to avoid invalid states. Such validation doesn't happen in a pure XML merge, which causes the risk of creating invalid models that might not even open.

If each team makes changes only to their diagrams, and if these changes represent classes in separate packages, the problem could be reduced, as most changes would appear in separate files. Even so, inevitably there will be some cases where relationships that cross package boundaries are changed.

There's no good story for automatic model merging yet.

In reality, some teams will want to branch when creating new product iterations, which causes forking of source code, documentation and models. Models such as the activity, sequence, layer and class diagram are good examples of models that evolve over iterations, while the delivery team continues with the mainstream development and maintenance. Therefore, models may and often *will* evolve in two or more branches, which means that we'll encounter the branching scenario and the often-challenging merge scenario at some point.

All of the current models are good candidates for branching, but none are conducive to merging. With the prospect of a challenging and error-prone merge, the recommendation is twofold:

- Avoid a merge by defining a solution and model view that represents classes in separate packages. The architecture tooling guidance proposes a solution view, shown in **Figure 8**, and a model view, shown in **Figure 9**, based on packages. Some care must be taken when diagrams have content from multiple packages, which is possible for Class, Component and Use Case. In this case, to fully avoid conflicts, users must avoid editing the metadata of elements that belong to the "foreign" package.
- Keep models on a branch that won't be forked, similar to shared components.

The fallback is to visually and manually edit the models in one branch, using the options "Keep Target Branch Version" or "Take Source Branch Version" options as shown in **Figure 10**.

For example, the models diverge in the two branches as shown and are merged manually (step 3) by visually comparing the models and manually updating the model in the top branch. The branch with the consolidated model is then reverse-integrated into Main (step 4) and the other branch with the outdated model is reverse-integrated (step 5) using "Take Target Branch Version" when resolving model conflicts.

In summary, there's no good story for automatic model merging yet. The recommended strategy is to avoid a branch and merge scenario with the models, or to use the visual and manual model editing before merging.

We've now introduced a number of new branching scenarios that you may encounter in a complex real-world environment. In the next article in this series, we'll investigate team projects and team project collections. ■

BILL HEYS *is a senior consultant with Microsoft Americas Consulting Services in New England. As a Visual Studio ALM Ranger, Heys specializes in custom application development and Application Lifecycle Management using Visual Studio. His blog is at blogs.msdn.com/b/billheys.*

WILLY-PETER SCHAUB *is a solution architect with the Visual Studio ALM Rangers at the Microsoft Canada Development Center. Since the mid-'80s he's been striving for simplicity and maintainability in software engineering. His blog is at blogs.msdn.com/b/willy-peter_schaub.*

THANKS to the following technical experts for reviewing this article: *Marcel de Vries, Jens Jacobsen, Bijan Javidi and Alan Wills*

Securing WF 4 Workflow Services

Zulfiqar Ahmed

Windows Workflow Foundation (WF) provides a visual authoring experience for writing software logic. Once software logic is implemented as a workflow, it's executed by hosting the workflow in a workflow host. A workflow service is a special type of Web service implemented using a workflow and is made available to the consumer by hosting it in a WorkflowServiceHost. In this article, I'll talk about security options of the different workflow hosts with a particular focus on the workflow services and the WorkflowServiceHost. I'll explain some key extensibility points that can be used to extend the workflow services security boundary to the workflow layer. I'll also discuss the Workflow Security Pack (WFSP) project and how its collection of activities can be used to bring end-to-end security to workflow solutions. WF 4, which is shipped as part of the Microsoft .NET Framework 4, provides an extensible hosting API and comes out-of-the-box with three different hosts with varying capabilities.

This article discusses a prerelease version of the Workflow Security Pack. All information is subject to change.

This article discusses:

- Workflow host options
- Workflow services wire security
- The OperationContext
- Workflow services and Windows Identity Foundation
- Workflow Security Pack Community Technology Preview 1
- In-workflow authorization
- Messaging activities and authenticated messaging
- Windows impersonation/delegation

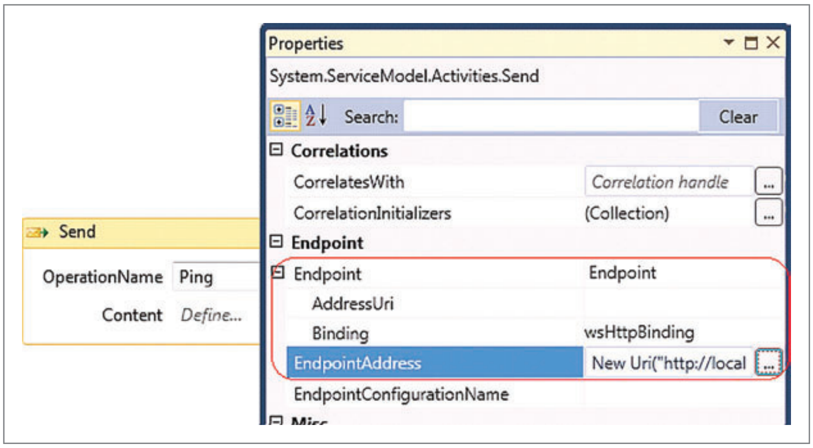
Technologies discussed:

Windows Workflow Foundation, Workflow Security Pack

Code download available at:

code.msdn.microsoft.com/mag201102Workflow

WorkflowInvoker This is the most basic and least capable host interface, providing a simple API for invoking workflows. A WorkflowInvoker object only supports a single workflow instance, passed to it via the constructor or the static Invoke method. All workflow execution is guaranteed to be on the same calling thread, so if calling code is impersonating a particular security context, all the activities will execute under this impersonated context. WorkflowInvoker isn't a Workflow Host in the true sense; rather, it encapsulates a WorkflowApplication-based host and uses a pump-based synchronization context to provide an easy-to-use API with consistent execution semantics. For example, Exceptions, Transactions and so on seamlessly flow across the invocation boundary. This behavior simplifies the security, as the security context of the caller is available throughout the workflow execution and activities can use it in various security scenarios. For example, Principal-Permission authorization and Kerberos-delegation work seamlessly with WorkflowInvoker.



Send

OperationName	Ping
Content	Define...

Properties

System.ServiceModel.Activities.Send

Search: Clear

⊟ Correlations	
CorrelatesWith	Correlation handle
CorrelationInitializers	(Collection)
⊟ Endpoint	
⊟ Endpoint	Endpoint
AddressUri	
Binding	wsHttpBinding
EndpointAddress	New Uri("http://local
EndpointConfigurationName	

Figure 1 **Send Activity Properties**

WorkflowApplication This is a slightly more capable host but still supports only a single instance. This host executes workflow using the IO threads from the CLR ThreadPool. The security context of a calling thread isn't copied to the workflow thread, so even if the workflow client is impersonating, the WF thread—which is executing the activities—won't be impersonating. The security context of the caller can be flown to the WF thread using a custom Synchronization context that would forward the call on the same incoming Async thread, similar to the Synchronization context used by the WorkflowInvoker:

```
public class MySyncContext : SynchronizationContext
{
  public override void Post(SendOrPostCallback d, object state)
  {
    d(state);
  }
}
```

WorkflowServiceHost This is the most comprehensive host, providing a hosting environment suitable for multiple workflow instances. Workflow services are a special type of Web services whose implementation is based on workflows. WorkflowServiceHost derives from the standard Windows Communication Foundation (WCF) ServiceHostBase class, and all the WCF security concepts apply to WorkflowServiceHost as well. Messaging activities are the primary interaction model supported by the WorkflowServiceHost, along with the WorkflowHostingEndpoint, which enables the use of WorkflowServiceHost without using messaging activities. In this article, I'll primarily focus on the security aspect of messaging activities, Workflow Services and the WorkflowServiceHost. For an overview of workflow services technology, please check out Leon Welicki's article, "Visual Design of Workflows with WCF and WF 4," in the May 2010 issue of *MSDN Magazine* (msdn.microsoft.com/magazine/ff646977).

Figure 2 **Infrastructure Messages Sent to Negotiate Service Credentials**

http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT
http://tempuri.org/IPingService/Ping
http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT/Cancel

Figure 3 **A Simple Scope Activity to Add Execution Properties**

```
[ContentProperty("Body")]
public sealed class OperationContextScope : NativeActivity
{
  public Activity Body { get; set; }
  protected override void Execute(NativeActivityContext context)
  {
    if (this.Body != null)
    {
      // Adding an execution property to handle OperationContext
      context.Properties.Add(OperationContextScopeProperty.Name,
        new OperationContextScopeProperty());
      context.ScheduleActivity(this.Body);
    }
  }
}
```

Workflow Services Wire Security

As workflow services are standard WCF services, wire security aspects are configured using the standard WCF binding mechanisms. A workflow service can be exposed using one or more endpoints using a particular binding as per the security needs of the service. A WCF dispatch pipeline will only execute if the incoming message satisfies the security requirements of the target endpoint. Workflow logic executes at the end of a dispatch pipeline, so all common WCF extensibility points are applicable to workflow services as well. For example, the standard ServiceAuthorizationManager extensibility point can be used to apply authorization for workflow services as well. Frameworks like Windows Identity Foundation (WIF) integrate with WCF at the dispatcher level, and these can transparently be used with workflow services as well. There are some threading

Figure 4 **ReceiveMessageCallback Implemented as an Execution Property**

```
[DataContract]
class OperationContextScopeProperty : IReceiveMessageCallback,
IExecutionProperty
{
  private OperationContext current;
  private OperationContext orignal;

  public static readonly string Name =
    typeof(OperationContextScopeProperty).FullName;
  public void OnReceiveMessage(OperationContext operationContext,
    ExecutionProperties activityExecutionProperties)
  {
    current = operationContext;
    operationContext.OperationCompleted
      += delegate(object sender, EventArgs e)
    {
      current = null;
    };
  }

  public void CleanupWorkflowThread()
  {
    OperationContext.Current = orignal;
  }
  public void SetupWorkflowThread()
  {
    orignal = OperationContext.Current;
    OperationContext.Current = current;
    }
}
```

differences, related to few async points between the WCF and WF layers, which make certain Thread Local Storage (TLS)-related scenarios a bit more challenging in the workflow services. For example, WIF exposes the incoming identity information using the Thread.CurrentPrincipal and it makes sure to set this correctly for the code-based services. However, your workflow logic might end up executing on a different thread than the original WCF thread. If that happens, all TLS-related data—including the Thread.CurrentPrincipal—won't be valid, so it's advised not to rely on TLS in your workflows. I'll talk about some potential solutions to this in a later section.

WF 4 also provides a Send activity for calling other Web services from within a workflow. The Send activity can be configured with a binding that would be used when calling other services from within the workflow. Internally, the Send activity uses the standard WCF ChannelFactory/Channel API for sending messages, and the configured binding is used to create this internal channel factory. Send activity also has a caching layer built into it that's used for ChannelFactory/Channel caching. By default, this caching layer is only used if the endpoint information is specified directly using the properties of the Send activity and a stock binding is chosen, as shown in **Figure 1**.

As soon as endpoint information is loaded from the config file using the EndpointConfigurationName property, the safe caching is disabled and every execution of Send activity creates a brand-new ChannelFactory. Secure bindings such as wsHttpBinding and wsFederationHttpBinding do quite a lot of work at the channel factory opening stage, and recreating a channel factory for each message could be quite expensive. For example, the default WSHttpBinding is optimized for performance and security and achieves this by establishing a secure conversation session that has an upfront cost, but subsequent messages can be secured with a much smaller cost. Without ChannelFactory caching, this optimum behavior of WSHttpBinding becomes an overhead because for every business message, four additional infrastructure messages are sent to negotiate the service credentials and then to establish a secure conversation session, as shown in **Figure 2**.

In WF 4, any binding configuration loaded from the configuration file (even the default bindings) is treated as "unsafe for caching" by the Send activity, and the ChannelFactory caching is disabled. This default behavior can be overridden by forcing unsafe channel caching, which will reuse the ChannelFactory for sending messages to the same endpoint. SendMessageChannelCache service behavior enables unsafe caching and also allows the configuration for various Channel and ChannelFactory cache settings, as shown here:

```
<serviceBehaviors>
  <behavior>
    <sendMessageChannelCache allowUnsafeCaching="true">
      <channelSettings idleTimeout="1:0:0" maxItemsInCache="60"/>
      <factorySettings idleTimeout="1:0:0" maxItemsInCache="60"/>
    </sendMessageChannelCache>
  </behavior>
</serviceBehaviors>
```

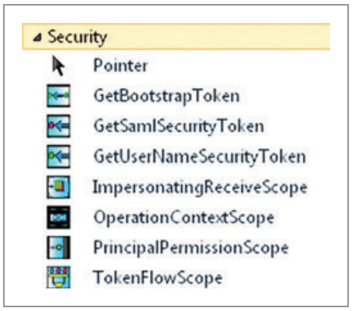

Security
Pointer
GetBootstrapToken
GetSamlSecurityToken
GetUserNameSecurityToken
ImpersonatingReceiveScope
OperationContextScope
PrincipalPermissionScope
TokenFlowScope

Figure 5 **Workflow Security Pack Activities**

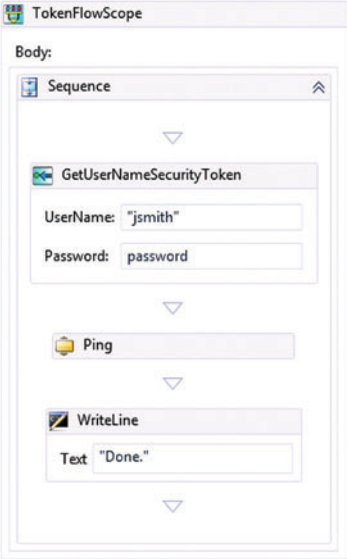

TokenFlowScope
Body:
Sequence
GetUserNameSecurityToken
UserName: "jsmith"
Password: password
Ping
WriteLine
Text "Done."

Figure 6 **Authenticated Messaging Using a UserName Token**

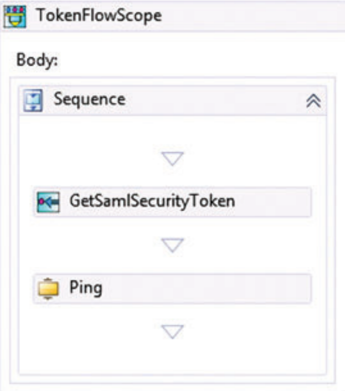

TokenFlowScope
Body:
Sequence
GetSamlSecurityToken
Ping

Figure 7 **Fine-Grained Control of SAML Token Acquisition and Usage**

Where Is My OperationContext?

In traditional code-based services, security information for the incoming call is available via the OperationContext.Current. The WCF dispatcher runtime makes sure to set the OperationContext on the thread just before calling the service method, so inside the service method, security information is accessible using the OperationContext.Current.

Workflow services add additional complexity because there are a bunch of async points between the WCF dispatcher and workflow execution. Any one of these async points can switch the thread, in which case workflow logic (activities) would execute on a different thread from the WCF thread and OperationContext wouldn't be available using the OperationContext.Current approach. In WF 4, thread-agnostic access to OperationContext is enabled using a callback mechanism based on IReceiveMessageCallback and ISendMessageCallback. IReceiveMessageCallbacks are used on the server side, while ISendMessageCallbacks give access to the OperationContext on the client side. On the server side, IReceiveMessageCallbacks are invoked just after a message is received by a Receive activity. To attach these callbacks to Send and Receive activities, they must be available as execution properties when Send/Receive executes. A common approach for adding execution properties to an activity is to create a parent scope activity and set the execution properties as part of the parent activity's execution, as shown in **Figure 3**.

In the code snippet in **Figure 3**, when the OperationContextScope activity executes, it simply adds an execution property to the context so that all the child activities can see this execution property. Send and Receive activities look for the previously mentioned callback properties, and if one of these properties is found, it's invoked at the correct stage of message processing, giving you access to the OperationContext. In this example, any Receive activity that would be part of the same scope sees the OperationContextScopeProperty and executes the callback passing in the OperationContext value (see **Figure 4**).

Figure 8 **WFSP Extensibility: Implementing Additional Token Types**

```
[Designer(typeof(GetX509SecurityTokenDesigner))]
public class GetX509SecurityToken : GetSecurityToken
{
  public GetX509SecurityToken()
  {
    FindType = X509FindType.FindBySubjectName;
    StoreLocation = StoreLocation.CurrentUser;
    StoreName = StoreName.My;
  }

  public InArgument<X509Certificate2> Certificate { get; set; }
  public X509FindType FindType { get; set; }
  public StoreLocation StoreLocation { get; set; }
  public InArgument<string> FindValue { get; set; }
  public StoreName StoreName { get; set; }

  protected override void Execute(NativeActivityContext context)
  {
    X509Certificate2 targetCert = null;
    if (this.Certificate != null)
      targetCert = this.Certificate.Get(context);
    if (targetCert == null)
    {
      var store = new X509Store(StoreName, StoreLocation);
      try
      {
        store.Open(OpenFlags.ReadOnly);
        var col = store.Certificates.Find(FindType, FindValue.Get(context), false);
        if (col.Count > 0)
          targetCert = col[0];//Use first certificate mathing the search criteria
      }
      finally
      {
        if (store != null)
          store.Close();
      }
    }
    if (targetCert == null)
      throw new InvalidOperationException(
        "No certificate found using the specified find criteria.");
        // Enlist the token as a flow token
      base.EnlistSecurityToken(context, new X509SecurityToken(targetCert));
  }
}
```

OperationContextScopeProperty simply captures and stores the currently active OperationContext and later on sets it on the correct WF thread using the WF TLS mechanism. The IExecutionProperty interface has Setup/CleanUpWorkflowThread methods, which are called before and after executing every WF work item (activity) and give the ability to set various TLS-related properties on the selected WF thread, OperationContext being one example in this case.

OperationContextScope is an example of a custom activity that leverages the WF 4 extensibility to enable thread-agnostic access to the WCF OperationContext for all in-scope child activities.

Workflow Services and WIF

WIF provides a rich API and object model to claim-enable WCF services and ASP.NET applications. WIF integrates with WCF at the host level, so most of the WIF features work with Workflow Services as well. Please check my blog post bit.ly/a6pWgA for additional details on integrating WIF with Workflow Services. The out-of-box integration works fine for basic scenarios, while additional rich scenarios can be enabled using the activities from WFSP.

Introducing WFSP CTP 1

WFSP Community Technology Preview (CTP) 1 provides a collection of activities and associated WCF behavior to enable key security scenarios in WF 4. WFSP leverages the ISend/IReceiveMessageCallback extensibility model to implement many of its features. CTP 1 of WFSP was released on CodePlex in July 2010 and can be downloaded from wf.codeplex.com/releases/view/48114.

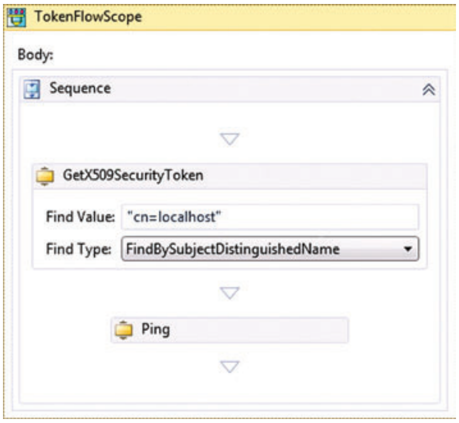
As workflow services are standard WCF services, wire security aspects are configured using the standard WCF binding mechanisms.

WFSP activities, shown in **Figure 5**, blend nicely with the rest of the WF and provide powerful constructs to bring integrated security in workflow solutions.

In-Workflow Authorization

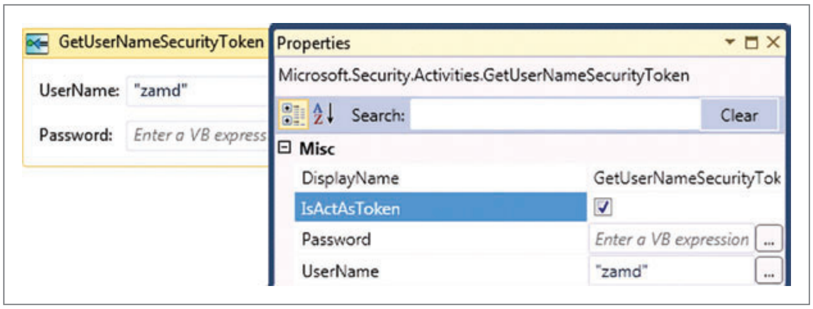
In Workflow Services, you can use the standard WCF ServiceAuthorizationManager extensibility to enforce authorization, and that feature works exactly the same as in code-based services. However, in some scenarios (for example, where authorization data is part of workflow), you'd like to delay the authorization decision until actual workflow execution. PrincipalPermissionScope is a server-side activity that brings the CLR PrincipalPermission authorization feature to the workflows. Any child activity placed inside the scope will only execute if the permission demand was successful. This activity looks for the identity information in the incoming WCF security context accessed from the OperationContext. PrincipalPermissionScope is implemented using the same IReceiveMessageCallback mechanism mentioned earlier in this article.

The actual PrincipalPermission demand is enforced against an IPrincipal object based on the value of ServiceAuthorizationBehavior.PrincipalPermissionMode. This extensibility feature enables PrincipalPermissionScope to work with an ASP.NET Role Provider as well as a custom IPrincipal implementation produced by a custom IAuthorizationPolicy. Check out msdn.microsoft.com/library/aa702542 for details on how to configure ASP.NET Role Provider with a WCF service.



Find Value: "cn=localhost"
Find Type: FindBySubjectDistinguishedName

Figure 9 **TokenFlowScope with a Custom GetToken Activity**



GetUserNameSecurityToken

UserName: "zamd"

Password: Enter a VB express

Properties

Microsoft.Security.Activities.GetUserNameSecurityToken

Search: Clear

⊟ Misc

DisplayName	GetUserNameSecurityTok
IsActAsToken	☑
Password	Enter a VB expression
UserName	"zamd"

Figure 10 **Creating an ActAs Token**

Messaging Activities and Authenticated Messaging

Send activity provides the primary way of consuming Web services from within workflows. In most real-world scenarios, these back-end services are secured and require authentication before doing any work. In standard code-based services, credential information is specified using the ClientCredential property exposed on a ChannelFactory and a ClientBase<T> derived proxy class. Using this property, a client can specify the credential it wishes to use before calling the service. Unfortunately, Send activity, which wraps the ChannelFactory, doesn't expose ClientCredential in WF 4, so some of the scenarios that require explicit credential specification aren't possible with out-of-box Send activity. Note that Send activity does pick the endpoint behavior configuration from the config file, so you can create a custom endpoint behavior to specify these credentials.

A typical example that requires explicit credentials is calling a service configured to require a UserName/Password. As the ClientCredential property isn't exposed on the Send activity, there's no way to specify a UserName/Password. Let's see how GetUserNameSecurityToken activity from WFSP provides a solution to this and other related scenarios.

In **Figure 6**, Ping activity is generated by the "Add Service Reference" wizard and is configured to call a service that's secured to require UserName authentication, as shown in following binding configuration:

```
<wsHttpBinding>
  <binding name="singleShotUserName">
    <security mode="Message">
      <message clientCredentialType="UserName" establishSecurityContext
        ="false" />
    </security>
  </binding>
</wsHttpBinding>
```

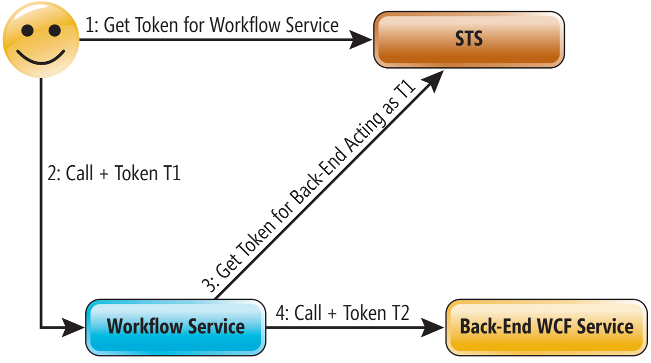


Figure 11 **End-to-End Claims-Based Delegation Flow**

In the preceding workflow, GetUserNameSecurityToken creates a UserNameSecurityToken based on the supplied UserName/Password and enlists it with the ambient SecurityTokenHandle provided by the TokenFlowScope activity. "Workflow Security Pack" applies security at the SecurityToken level as opposed to the ChannelFactory /ClientBase<T> approach of applying security at the credential level. In standard WCF, credentials are used to create security tokens, but WFSP uses security tokens directly and taps into the WCF security layer at the token level rather than the credentials level.

TokenFlowScope is the key activity that enables authenticated messaging and other interesting scenarios. This activity, along with the WorkflowClientCredentials endpoint behavior, flows the enlisted tokens from the workflow layer to the WCF security layer, where they're attached with the outgoing message as per the binding requirements of the endpoint. TokenFlowScope requires a custom ClientCredential behavior (WorkflowClientCredentials) to be configured, as shown in the following configuration snippet:

```
<behavior>
  <!--This custom clientCredentials enables the credential flow from
    workflow data model into WCF security layer. -->
  <clientCredentials
    type="Microsoft.Security.Activities.WorkflowClientCredentials,
    Microsoft.Security.Activities, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=31bf3856ad364e35">
  </clientCredentials>
</behavior>
```

WFSP follows this exact model when calling a service that requires a token from a Security Token Service (STS), as shown in **Figure 7**.

Windows Identity Foundation provides a rich API and object model to claim-enable WCF services and ASP.NET applications.

In **Figure 7**, GetSamlSecurityToken goes to an issuer and acquires a SAML token that's then enlisted with the ambient handle provided by the TokenFlowScope activity. This enlistment makes this token available to any Send activity living in the same scope and requiring a SAML token. The model is extensible, and GetSamlSecurityToken can itself use an already enlisted token while acquiring a SAML token, for example, if the STS requires a UserName token to return a SAML token and if there's already a valid UserName token enlisted in the scope. GetSamlSecurityToken, when configured with WorkflowClientCredentials behavior, would use this token when requesting a SAML token.

Out-of-box WFSP only supports UserName and SAML token types; however, other token types can be enabled by inheriting from the GetSecurityToken class as shown in the code snippet in **Figure 8**, which implements an activity to create an X509-based token.

GetX509SecurityToken creates an X509-Security token based on a certificate and enlists it with the SecurityTokenHandle as a flow token that can then be used to call a service requiring a certificate for authentication. **Figure 9** shows GetX509SecurityToken in use with a custom activity designer.

Claims-Based Delegation

Claims-based delegation is another useful feature enabled by WFSP. Claims-based delegation is often more desirable for workflow services, as these services primarily implement an orchestration/business process calling multiple back-end services. Additionally, access to a caller's identity in those back-end services is often required to enable fine-grained authorization decisions. WFSP leverages the ActAs functionality of WS-Trust 1.4 to enable any token type to be used as an ActAs token. By default, all the GetToken activities create a token and enlist it as a flow token—however, all of these activities also have a flag known as IsActAsToken, as shown in **Figure 10**.

When this flag is checked, the token-creation logic stays the same, but the created token T1 is enlisted as an ActAs token rather than a flow token. There can be only one ActAs token per scope, and it's consumed by GetSamlSecurityToken activity when requesting a SAML token. When GetSamlSecurityToken executes, the active ActAs token is picked up and is sent as part of a token-issuance request generated by the GetSamlSecurityToken activity. The returned token T2 would contain claims from both the authentication token as well as the ActAs token. Finally, any Send activity executing inside this scope can use this T2 token when calling a back-end service that would see both the identities in its security context.

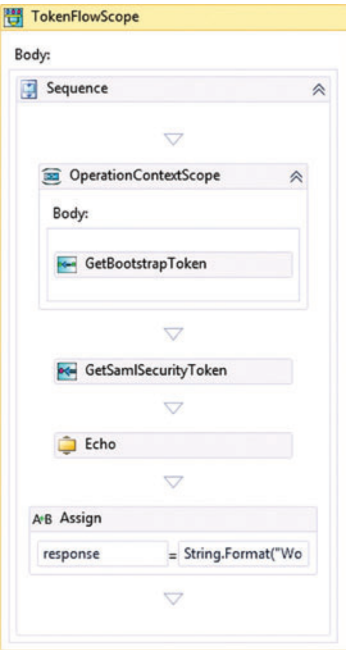


Figure 12 **Claims-Based Delegation Workflow**

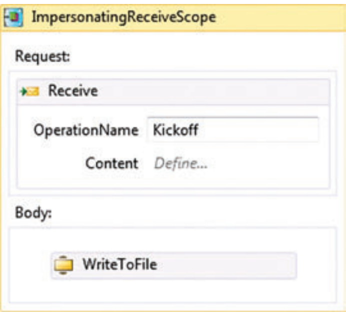


Figure 13 **ImpersonatingRecieve-Scope in Action**

GetBootstrapToken activity is used in middle-tier scenarios to enable end-to-end claims-based delegation. As opposed to GetToken activities, this activity simply reads the incoming token and enlists it as an ActAs token rather than creating a new token and then enlisting it. GetBootstrapToken activity enables a workflow service to use the identity of the incoming caller, in addition to its own identity, when calling back-end services, as shown in **Figure 11**.

In Step 3 of **Figure 11**, workflow service uses WFSP activities to read the incoming bootstrap token, acquires a new token acting as the bootstrap token identity and then flows both identities to a back-end server. **Figure 12** shows the workflow that's used by this workflow service.

In the workflow shown in Figure 12, GetBootstrap activity is placed inside an OperationContextScope to guarantee thread-agnostic access to an OperationContext when this activity executes. GetSamlSecurityToken uses the ActAs token produced in the previous step by the GetBootstrapToken activity and then, finally, Echo activity calls the back-end service with a final SAML token produced by the GetSamlSecurityToken activity.

Windows Impersonation/Delegation

ImpersonatingReceiveScope is another server-side activity that brings Windows impersonation and delegation to the workflow world. When this activity executes, it looks for a WindowsIdentity inside the incoming security context. If the incoming message produces a WindowsIdentity, all the child activities that are part of the body will execute inside this impersonated scope. ImpersonatingReceiveScope uses the Workflow TLS mechanism, mentioned earlier in this article, to impersonate the identity on the WF thread just before executing a work item. Impersonation is reverted when the WF work item completes execution.

Failing to find a valid WindowsIdentity in the incoming security context, ImpersonatingReceiveScope looks for a UPN claim—either in a WIF identity (Thread.CurrentPrincipal) or in the traditional WCF ClaimsSet—and uses it to create a WindowsToken using the S4U features of Kerberos. To transform a UPN claim to a Windows token, ImpersonatingReceiveScope relies on "Claims to Windows Token Service," which is part of the WIF runtime. This service must be installed and running for the claim-to-token transformation to be successful.

Figure 13 shows a typical use of ImpersonatingReceiveScope activity.

End-to-End Security

From the outside, workflow services are standard WCF services, and as such, most of the WCF security options are applicable to workflow services as well. WF 4 introduced a couple of key extensibility points that can be used to further extend the workflow services security boundary to the workflow layer. WFSP provides a collection of activities that use these extensibility points to bring end-to-end security to WF 4. ■

ZULFIQAR AHMED *is a senior consultant on the Premier Support for Developers team and is the original author of the Workflow Security Pack project. He maintains a blog at zamd.net, where he covers topics ranging from Windows Communication Foundation, Windows Workflow Foundation and claims-based security to general Microsoft .NET Framework stuff.*

THANKS to the following technical expert for reviewing this article: *Dave Cliffe*

Measuring Test Effort Progress with EVM

In many software development environments, an important part of the overall software-testing effort is having the ability to measure the progress of the test effort. One technique for doing this is a project management technique called Earned Value Management (EVM).

EVM is a simple quantitative technique that can be used to measure the schedule progress—and optionally the budget progress—of any type of project, including a software-testing effort or some part of the overall effort. EVM had its origins in a 1962 initiative by the U.S. Department of Defense called PERT/Cost. Although EVM is simple to use and can be applied to test efforts of any size, based on my experience many software engineers incorrectly believe that EVM is suitable for use only with large software development efforts.

In this month's Test Run column I explain what EVM is, walk you through an example of using EVM to measure test-effort progress, and describe when to use EVM and when not to use it.

> Break down the part of the testing effort you want to monitor into smaller tasks.

Preparing Earned Value Management

The best way to understand EVM is to walk through a concrete example. The first step in EVM is to break down the part of the testing effort you want to monitor into smaller tasks. In standard project management terminology these smaller tasks are usually called work packages, but in a software development environment, often they're simply called subtasks.

Let me emphasize up front that breaking down a software testing effort, or any project for that matter, typically is the most difficult part of the EVM process.

Now let's assume that you break down your test effort into five subtasks, labeled A through E, as shown in **Figure 1**. The level of detail, or granularity, you break your overall testing effort down to depends on many factors. As a general rule of thumb, in a software-testing environment, individual subtasks are often scoped so that each subtask requires roughly between four and 40 hours to complete.

The topology of the diagram in **Figure 1** indicates that testing subtask A must be completed before B starts, and that subtasks C and D must both be completed before subtask E can start.

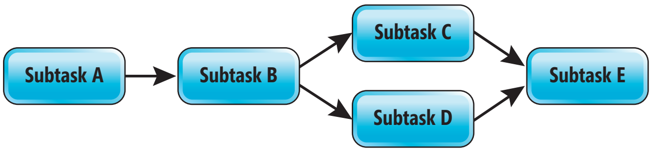


Figure 1 **Mapping Subtasks for EVM**

The next step in EVM is to estimate how much of the overall test resource allotment, or budget, is associated with each subtask. This is called the planned value (PV) for each subtask.

The example in **Figure 2** assumes that you have a total test budget of 350 units. PV units are most often measured in dollars (or Euros or rubles and so on). The absolute magnitude of cost units is unimportant, so the 350 total planned value in our example could represent $350 or $350,000. Besides monetary units such as dollars, the units of PV can also be arbitrary units that measure the cost in some way.

There are no magic formulas to determine the subtask PV, but it's important to note that the accuracy of the schedule progress metrics produced by EVM depends entirely on the accuracy of your initial PV metrics. In **Figure 2**, you can see the PV estimates for subtasks A, B, C, D and E are 50, 60, 90, 80 and 70 respectively.

After estimating PV, the next step in EVM is to estimate how long each subtask will take, and then use these estimates to determine the finish time for each subtask. In **Figure 2**, I determined (using historical data, or previous experience, or some quantitative technique) that subtasks A, B, C, D and E are estimated to take 1, 2, 3, 1 and 2 days each. In this example, the units of time are days, but you can use hours or weeks or any measure as long as you're consistent throughout the EVM analysis.

We start at time = 0, so if subtask A is estimated to take 1 day, it will finish at day 1. Subtask B would then begin at day 1 and require 2 days, ending at day 3. Notice that subtask E cannot begin until subtasks C and D both finish, which would be on day 6 (the larger of the finish times for C and D), and so would end at day 8.

Figure 2 **Scheduling Subtasks**

Subtask	Planned Value	Duration (days)	Finish (days)
A	50	1	1
B	60	2	3
C	90	3	6
D	80	1	4
E	70	2	8

Once you've determined what your software test effort subtasks are, and estimated their PV, durations and finish times, the next step is to create a table of cumulative PVs. You begin by constructing a table like the one shown in **Figure 3**. The leftmost column marks the end of each unit of time (in this example, days 1 through 8). The second column is the cumulative PV at the end of each day, which can be determined from the previous table of PV data.

At the end of day 1, subtask A should be completed and so the cumulative PV should be 50, the PV for A. At the end of day 2, no new subtasks are expected to be finished so the cumulative PV is still 50. At the end of day 3, subtask B should now be finished and so the cumulative PV should be 50 for subtask A plus 60 for subtask B = 110. In the same way the cumulative PVs at the end of days 4 through 8 can be determined.

There are no magic formulas to determine subtask planned value.

Measuring Test Schedule Progress

Let's suppose that your test effort unfolds as shown in the third column in the table in **Figure 3**. These activities represent what actually happens as opposed to what you expect to happen. The difference between what you had planned to accomplish (your PV) and what you actually accomplished is your earned value (EV).

So, at the end of day 1, subtask A started but did not finish as scheduled. Therefore the cumulative EV in column 4 is 0. At the end of day 2, subtask A does in fact finish and so I earn 50 (the PV associated with A), and place that value in column 4. At the end of day 3, subtask B finishes and so the cumulative EV is 50 + 60 = 110. But at the end of day 4, no new subtasks finish and so the cumulative EV is still 110. At the end of each day, or whatever time unit you're using, you update the cumulative EV column.

Your test effort schedule progress is easily read from the table in **Figure 3**. If the cumulative EV is less than the cumulative PV, then you're behind schedule. If the cumulative EV is exactly equal to the cumulative PV, then you're on schedule. And if the cumulative EV is greater than the cumulative PV, then you're ahead of schedule (which, by the way, is not necessarily always a good thing).

EVM typically uses two specific metrics to quantify how much ahead, behind or on schedule your test effort is. The so-called schedule variance (SV) at any given point in time is simply the cumulative EV minus the cumulative PV. For example, in **Figure 3**, at the end of day 4, SV = 110 – 190 = -80 indicating the test effort is 80 cost units (typically dollars) of PV behind schedule. A negative SV indicates a project is behind schedule and positive SV indicates a project is ahead of schedule.

Figure 3 **Schedule Progress with Cumulative PV**

Day	Cum PV	Actual Activity	Cum EV
1	50	A has started (but has not yet finished)	0
2	50	A has finished, B has started	50
3	110	B has finished, C and D have started	110
4	190	(no change)	110
5	190	(no change)	110
6	280	Both C and D have finished, E has started	280
7	280	(no change)	280
8	350	E has finished	280

Because the absolute magnitude of SV depends on the units of PV, an alternative metric called the schedule performance index (SPI) is often used instead of SV. SPI is cumulative EV divided by PV. In my example, at the end of day 4, the SPI is 110 / 190 = 0.58.

This can be interpreted to mean that I have only earned 58 percent of my PV—in other words, I'm 42 percent behind my scheduled PV.

SPI values less than 1.00 mean the test effort is behind schedule, an SPI value of 1.00 means the test effort is exactly on schedule, and SPI values greater than 1.00 mean the test effort is ahead of schedule.

Wrapping Up

As you've seen in this column, monitoring software-test effort schedule progress with EVM is easy. However, as with any quantitative technique, your results are only as good as your initial data—in this case the PVs associated with each test effort subtask. EVM is a dynamic activity and you should revise your estimates as your test effort unfolds.

In the introduction to this column I mentioned that EVM can be used to measure both schedule progress and budget progress. Measuring schedule progress as I've explained here is a prerequisite for measuring budget progress. Measuring budget progress requires that you actively monitor how much of your resources you spend at the end of each time unit. This is typically more difficult than measuring your work progress, so measuring budget progress with EVM is often only used on larger software projects and will be the subject of a future Test Run column.

As with any quantitative technique, your results are only as good as your initial data.

The EVM approach for measuring test effort schedule progress I've presented here, which essentially is a paper-and-pencil technique, is well suited for small projects and projects being developed in an Agile environment. When developing large software projects, software tools are usually needed to manage the complexity introduced by the existence of hundreds or even thousands of test effort subtasks. ■

Dr. James McCaffrey *works for Volt Information Sciences Inc., where he manages technical training for software engineers working at the Microsoft Redmond, Wash., campus. He's worked on several Microsoft products, including Internet Explorer and MSN Search. Dr. McCaffrey is the author of ".NET Test Automation Recipes" (Apress, 2006) and can be reached at jammc@microsoft.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *James Oker*

Sound Recording in Windows Phone 7

In one of the very first print advertisements for the introduction of the Macintosh in 1984, Apple touted the design of its mouse with an exceptionally compelling observation: "Some mice have two buttons. Macintosh has one. So it's extremely difficult to push the wrong button."

This isn't entirely true, of course. Overloading a single button with multiple functions can be just as confusing as multiple buttons. But the impossibility of pushing the wrong button is certainly a persuasive argument for simplicity in UI design.

Stripping down the UI to essentials is even more important when programming for a smartphone. Phones aren't very large. They simply can't have a lot of buttons, and the fingers that push these buttons are not as accurate as a mouse. Too many buttons means that it's easier than ever to push the wrong one.

On the minus side, limiting a UI often limits the functionality of the program, so deciding where to draw the line can be a real struggle. Life is full of compromises.

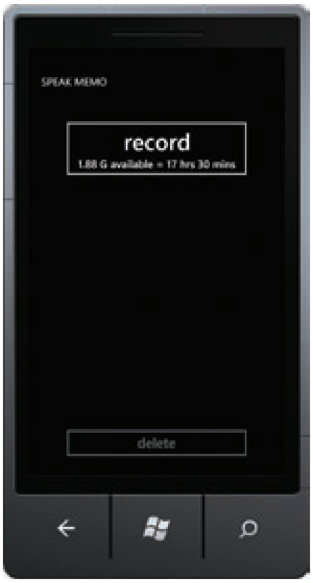


Figure 1 **The Initial SpeakMemo Screen**

Design Evolution

I thought it would be fun to write a Windows Phone 7 program that allows recording short vocal memos, such as "Remember to pick up the dry cleaning" and "Had a great idea for a movie: Boy meets girl."

Such a program is useful, of course, and provides yet another excuse to show off our new Windows Phones by using them in public places. More importantly for myself, I thought it would be a great opportunity to get some hands-on experience using the sound recording and playback classes supported by the phone.

However, the program design turned out to be more problematic than I had anticipated. Even before I'd written a single line of code, the program went through several iterations of design and redesign in my head.

At first, I thought it would be fine to have just two buttons labeled Record and Play, both of which functioned as toggles. Press the Record button to start recording and press it again to stop. The program saves the audio data in isolated storage. Press the Play button to play it back. Each press of the Record button replaces the previous memo so the program doesn't need a Delete button.

I even toyed around with reducing the program to just a Play button by implementing a voice-activation feature! The program would record continuously and only save the data when it contained some sounds. But this seemed like a devilishly difficult job in differentiating background sounds from real voice data without introducing some kind of manual threshold setting. I abandoned the single-button design.

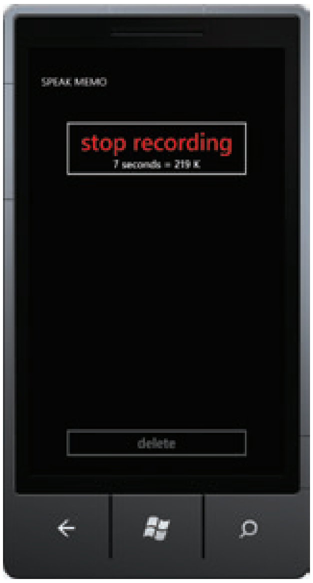
My original plan was fine for one memo, but not for multiple memos. I then thought that the program would maintain a single audio file and tack each new memo on the end of the previous memos. Because it's all just one big file, the Play button would play back all the memos in sequence. Of course, the program can't let this file grow indefinitely, so this design definitely needs a Delete button that wipes out the entire file and, consequently, all memos.

No, this wasn't good. I really needed to maintain separate files for each memo and allow these memos to be deleted individually. But that implied some way to present all the separate files to the user for playback and deletion, and all of a sudden the program got much more complex. I definitely needed a ListBox, and some way to identify each memo to the user, perhaps with user-supplied keywords or—horrors upon horrors—an actual file name.

Phones aren't very large. They simply can't have a lot of buttons.

No, no, no, not that! I glanced over at my telephone answering machine. Each call or memo is recorded separately, but they're numbered on a simple display. The Play button is complemented with Previous and Next transport buttons to go to the previous call or next call. As each memo or call is deleted, however, they

Code download available at code.msdn.microsoft.com/mag201102UIFrontiers.

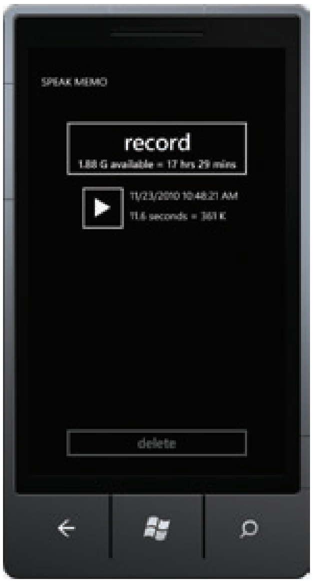


SPEAK MEMO

stop recording
7 seconds = 219 K

delete

Figure 2 **SpeakMemo While Recording**



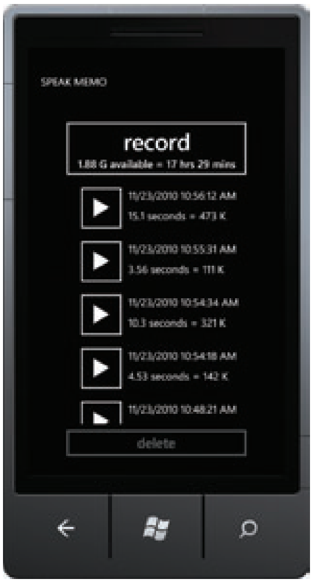
SPEAK MEMO

record
1.88 G available = 17 hrs 29 mins

11/23/2010 10:48:21 AM
11.6 seconds = 361 K

delete

Figure 3 **SpeakMemo with One Memo**



SPEAK MEMO

record
1.88 G available = 17 hrs 29 mins

11/23/2010 10:56:12 AM
15.1 seconds = 473 K

11/23/2010 10:55:31 AM
3.56 seconds = 111 K

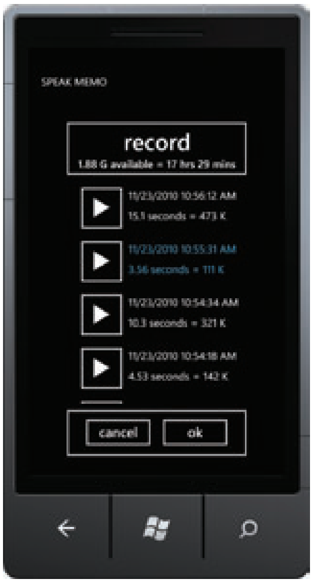
11/23/2010 10:54:34 AM
10.3 seconds = 321 K

11/23/2010 10:54:18 AM
4.53 seconds = 142 K

11/23/2010 10:48:21 AM

delete

Figure 4 **The SpeakMemo ListBox**



SPEAK MEMO

record
1.88 G available = 17 hrs 29 mins

11/23/2010 10:56:12 AM
15.1 seconds = 473 K

11/23/2010 10:55:31 AM
3.56 seconds = 111 K

11/23/2010 10:54:34 AM
10.3 seconds = 321 K

11/23/2010 10:54:18 AM
4.53 seconds = 142 K

cancel ok

Figure 5 **Confirming a Delete**

are renumbered. I knew I didn't want to number the memos, but I could take advantage of the larger display on the phone to show more detail about each one, including the record date, the duration and the file size.

The real breakthrough came when I realized I could put the ListBox on the program's main screen and use it not only for selection, but for playback as well.

Using the Program

My final design was, of course, a compromise between ultimate simplicity and a complete memo-management system. The downloadable SpeakMemo project is written for Silverlight for Windows Phone and requires the Windows Phone 7 Development Tools. You can run the program on the phone emulator, and it will appear to be working fine, but it won't actually record or play back any sounds.

The first time you run the SpeakMemo program, it displays the screen shown in **Figure 1**.

One button! Or, at least one *enabled* button on a fairly uncluttered screen. The button shows how much space exists in isolated storage and how that corresponds with a recorded sound file. (No, the program will not allow you to record a memo 17 hours in length!)

Press the Record button and it changes to a flashing red display with an updating duration indicator, as shown in **Figure 2**.

Press the Record button again, and the recorded memo shows up on the screen with the date and time, duration, storage space and Play button, as shown in **Figure 3**.

Of course, you can press the Play button to play it back, and the button toggles between Play and Pause modes.

It might not be so obvious with only one memo, but the recorded memos are stored in a ListBox in reverse chronological order as shown in **Figure 4**, so as you accumulate many of them, you can scroll through and play them individually.

One of the powerful features of Silverlight is the DataTemplate that lets you define the appearance of items in a ListBox. This DataTemplate can include other controls, such as buttons. I was pleased to come up with a practical application of putting a Button in a DataTemplate.

You can also manage the collected memos by deleting individual ones. When a memo is selected, the Delete button is enabled. Perhaps inspired by putting a Button in a DataTemplate, I performed another Silverlight trick by putting two additional buttons inside the Delete button. These buttons become visible when you press Delete, and they perform the traditional confirmation function, as shown in **Figure 5**.

One button! Or, at least one enabled button on a fairly uncluttered screen.

Playing a memo causes it to be selected, but an item is not played when you select it by pressing on the area to the right of the Play button. The program lets you play one memo, record another and delete still another all at the same time.

The Phone and Sound

At one time Windows Phone 7 was supposed to have some of the speech recognition and synthesis support found in the Microsoft .NET Framework System.Speech namespaces. Perhaps you'll see that support in the future.

Until then, you can capture sound from the phone's microphone and play it back through the phone's speaker using classes in the Microsoft.Xna.Framework.Audio namespace. These are XNA classes, but you can also use them in Silverlight programs. To use XNA classes in a Silverlight project, simply add a reference to

Microsoft.Xna.Framework.dll to the project's references and ignore the warning message.

The classes in the Microsoft.Xna.Framework.Audio namespace are entirely separate from those in the Microsoft.Xna.Framework.Media namespace. The Media namespace contains classes for playing music from the phone's music library, which are compressed audio files in MP3 or WMA format that become objects of type Song. I show how to access the music library in Chapter 18 of my book, "Programming Windows Phone 7" (Microsoft Press, 2010), which can be downloaded for free from bit.ly/dr0Hdz. In a blog entry on my Web site, I also demonstrate how to play MP3 or WMA files that are stored within the program itself, or which are downloadable over the Internet (bit.ly/ea73Fz).

A program that requires converting text to speech will probably use a Web service.

In contrast, classes in the Microsoft.Xna.Framework.Audio namespace work with uncompressed audio data in the standard PCM format, which is the same method used for audio CDs and Windows WAV files. With PCM, the analog sound amplitude is sampled at a uniform rate (usually in the range of 8,000 to 48,000 samples per second) and each sample is usually stored as an 8-bit or 16-bit value. The storage required for a particular sound is the product of the duration in seconds, the sample rate and the number of bytes per sample (multiply by two for stereo).

If you need speech-recognition support in your Windows Phone 7 application you must provide it yourself, most likely via a Web service. Similarly, a program that requires converting text to speech will probably use a Web service, or wait until the phone provides that support. The Microsoft Translator app for Windows Phone does this using the Microsoft Translator service (microsofttranslator.com). The code and documentation for the Translator Starter Kit is being released on MSDN (msdn.microsoft.com/library/gg521144(VS.92).aspx) and AppHub (create.msdn.com/education/catalog/sample/translatorstarterkit).

When using XNA audio services, a Silverlight program must call the static FrameworkDispatcher.Update method at approximately the same rate as the video refresh rate, which on Windows Phone 7 is approximately 30 times a second. There's a description of how to do this in the article "Enable XNA Framework Events in Windows Phone Applications" within the XNA online documentation (msdn.microsoft.com/library/ff842408). In SpeakMemo, the XnaFrameworkDispatcherService class handles this job. This class is instantiated in the App.xaml file.

Sound Recording

To record sound through the phone's microphone, you use the Microphone class. You'll probably create an instance of this class with the static Default property:

```
Microphone microphone = Microphone.Default;
```

Alternatively, the static All property provides a collection of Microphone objects, but then you'll probably want to present the list to the user to select one.

The sample rate is fixed, cannot be changed and is reported by the SampleRate property to be 16,000 samples per second. According to the Nyquist sample theorem, this is suitable for recording sounds up to 8,000 Hz in frequency. This is fine for voice, but don't expect great results with music. Each sample is 2 bytes wide and monaural, which means that each second of recorded sound requires 32,000 bytes, and each minute is 1.9MB.

Microphone data is delivered to your program in buffers that are simply byte arrays. You'll install a handler for the BufferReader event and then call Start to start recording. When the Microphone object fires the BufferReady event, your code calls GetData with a byte array. On return from GetData, the buffer has been filled with PCM data. When your program wants to stop recording, call GetData once more to get the last partial buffer. The method returns the number of bytes transferred to the array. Then call Stop.

The only option that Microphone allows you is specifying the byte size of the buffer that you pass to GetData. The BufferSize property is a TimeSpan value that must be between 100 ms and 1,000 ms (one second) in increments of 10 ms. In SpeakMemo, I left it at the default value of 1,000.

For your convenience, the Microphone class has two methods to convert between buffer sizes and time. Unfortunately these methods are a little confusing because the names refer to "sample." The GetSampleDuration method basically divides a byte size by 32,000 and returns a TimeSpan indicating that many seconds. GetSampleSizeInBytes multiplies a TimeSpan duration in seconds by 32,000.

When SpeakMemo is recording, it accumulates multiple 32,000-byte buffers in a generic List collection. When recording stops, the program saves all the individual buffers to a file in isolated storage.

The program lets you play one memo, record another and delete still another all at the same time.

Once I decided that I wouldn't include a key-word feature to identify memos, I wanted the file to contain only the PCM data and not any supplementary information. However, I was quite startled to realize that the IsolatedStorageFile class in Silverlight for Windows Phone does not support the methods for accessing the file creation time or last write time, and I felt this information was crucial from the user's perspective.

This meant that the file name itself would have to include the date and time. I first tried creating a file name from a DateTime object using the "s" and "u" formatting options, but that didn't work. (Why it doesn't work I'll leave as a simple exercise for the reader.) I then fabricated my own file name string by piecing the various components of the date and time together.

XNA Sound Playback

The Microsoft.Xna.Framework.Audio namespace lets you play back pre-recorded sounds using the related SoundEffect and SoundEffectInstance classes, whose names surely betray their common function in the context of an XNA game! But the static SoundEffect.FromStream method requires a Stream object referencing a standard Windows WAV file complete with RIFF header, and I didn't want to bother with file formats.

Microphone data is delivered to your program in buffers that are simply byte arrays.

For working with raw PCM data rather than WAV files, you'll instead want to use the DynamicSoundEffectInstance class, which derives from SoundEffectInstance. This class is ideal for the data generated from the Microphone class or for programs that dynamically create their own waveform data, such as music synthesizer programs.

The DynamicSoundEffectInstance constructor requires a sample rate and a number of channels; if you're using this class with data generated from the microphone, obviously you'll want to keep it consistent:

```
DynamicSoundEffectInstance playback =
  new DynamicSoundEffectInstance(
  microphone.SampleRate, AudioChannels.Mono);
```

On the other hand, if you want the playback to sound like a fast-talking chipmunk, simply multiply that first argument by two. DynamicSoundEffectInstance expects data to have a 16-bit sample size. The class has Play, Pause, Resume and Stop methods to control the playback, and a State property indicates the current state. The class works somewhat the opposite of Microphone: It fires a BufferNeeded event when it requires a new buffer. Your job is to fill up a buffer with PCM data and call SubmitBuffer.

SpeakMemo stores recorded memos in isolated storage.

To avoid audible gaps in the sound, in the general case you'll want to maintain a queue of buffers in the DynamicSoundEffectInstance class and submit a new buffer while the previous buffer is still playing. The class helps out with a PendingBufferCount property that indicates the number of buffers in the queue. The BufferNeeded event is fired when the PendingBufferCount changes and is less than or equal to two.

However, if you just need to play an entire chunk of PCM data, it's possible to call SubmitBuffer without bothering with the BufferNeeded event. At first, this was how I was using the class in the SpeakMemo program, but I discovered it wasn't possible to determine when the buffer had completed playing. There is no "state changed" event, and even if there were, DynamicSoundEffectInstance doesn't switch from the Play state to the Stop state when finished with the buffer. It's still expecting more buffers. Not knowing this information prevented the program from correctly toggling the visuals of the Play/Pause button.

I ended up handling the BufferNeeded event, but only to take the opportunity to check the PendingBufferCount property. When PendingBufferCount gets down to zero, the buffer has completed playback.

Storage Issues

SpeakMemo stores recorded memos in isolated storage. Conceptually isolated storage is private to the application, but physically it's part of a total storage area that's analogous to the hard drive of a desktop computer. All the application executables are stored there, as well as the phone's photo library, music library, video library and much more. The hardware specification for Windows Phone 7 requires the phone to have at least 8GB of flash memory for this storage area, and the phone itself will alert the user when the storage is getting low.

Storing the memo files was not my big concern. I was more worried about the program's heap. Aside from the flash memory storage, the Windows Phone 7 hardware specification also requires 256MB of RAM. This is the memory that an application occupies when it's running, and which provides the program's local heap. My experimentations revealed that SpeakMemo could allocate an array up to 90MB in size before it raised an out-of-memory exception. This is equivalent to about 47 minutes of sound from the microphone.

This doesn't mean that a Windows Phone 7 program is necessarily limited to 47 minutes of recording time. But a program that wants to record that much continuous sound must progressively save buffers to isolated storage to free up memory, and then load the file incrementally when playing it back. This was not how SpeakMemo was structured. Instead, the program saved and loaded entire files, and I didn't feel inclined to abandon that much simpler structure.

For that reason, I simply set a 10-minute maximum on the memo duration. Once a recording reaches that length, it's simply stopped and saved (which itself requires several seconds). To keep the program simple, there's no warning. The recording simply stops as if the user had pressed the button. This automatic stop-and-save also occurs when the program is terminated or otherwise deactivated; for example, during tombstoning.

Of course, playing back a 10-minute memo is not exactly convenient, either. The Play button toggles between play and pause mode but there's no way to rewind or fast forward. Those features could be added, but you know what that requires, right?

Yes: more buttons. Or perhaps even a Slider. ■

CHARLES PETZOLD *is a longtime contributing editor to* MSDN Magazine. *His new book, "Programming Windows Phone 7" (Microsoft Press, 2010) is available as a free download from bit.ly/dr0Hdz.*

THANKS to the following technical expert for reviewing this article: *Mark Hopkins*

Never, Never Land

In last month's column, I mused about how, in software as in life itself, there's usually a time for an action and also a time for its opposite. This month, I want to discuss the opposite of that thought, by which I mean events for which there is never a time.

I often compare our software industry to the medical industry. A doctor can't completely control a patient's outcome. Medical situations vary, risks always exist, stuff happens. But certain patient-harming events should never, ever occur. We know what causes them, we know how to prevent them; therefore, their occurrence always constitutes malpractice. Reading the list of these "never events" (bit.ly/h9RMI8) makes you wince: operating on the wrong patient, or on the wrong part of the correct patient; leaving surgical instruments inside the patient and so on. My all-time favorite good news/bad news joke—"The bad news is that we amputated the wrong leg. The good news is that your other leg is getting better after all"—brutally illustrates the unacceptability of these events. (I know: "Plattski, you are one sick puppy." It's been said before.)

We need to adopt this same idea for our software: that certain occurrences are never, ever, acceptable. We need to define these events, publicize them and educate developers about what they are and how to avoid them. And we need to explain to users that they should never have to tolerate this behavior from their software and shouldn't be asked to.

Here's my first proposed never event for software. We wish our programs wouldn't crash, as doctors wish their patients wouldn't die (and they envy our reset buttons), but neither is going to happen anytime soon. Because we know that our programs will occasionally crash, I say that losing a user's work in a crash is a never event. Remember how you'd work in Word or Excel for two hours, then up would pop the dreaded Unrecoverable Application Error box and it was all gone? Not acceptable. Ever. No matter what.

I hear lazy geeks objecting. "That's not our problem, it's a matter of education. Users just have to save their work every 10 seconds, then they'll never lose anything." Balderdash. That's not the user's job, any more than it's the patient's job to tell the surgeon: "No, you dimwit, it's my other arm. Are you sure you remember which end of the scalpel to hold?" It's the surgeon's job to get the operation right, as it is ours to get the software right.

Because these events should never occur, it's a big story when they do. Consider respected surgeon David Ring, who operated on the wrong hand of one of his patients at Massachusetts General Hospital (and performed the wrong procedure, too). Rather than cover it up, or discuss it only in a closed mortality and morbidity conference, he published his own case in the prestigious New England Journal of Medicine (bit.ly/gzWN9q). The surgical team performed a full failure analysis to find the root cause. (It's far more complicated than you might think; read the article.) They reviewed protocols and changed some of them: for example, the alcohol prep that washed away the supposedly indelible surgical site markings was discontinued. The world is a better place for this intolerance of unacceptable events, and for the openness in dealing with those that manage to occur.

> Certain occurrences are never, ever, acceptable. We need to define these events, publicize them and educate developers about what they are and how to avoid them.

Our industry needs the same thing. Why was data lost? It shouldn't have been. Was the disk full? That's a capacity problem—we know how to solve that. Because some dimwit yanked the plug out of the wall? That's a durability problem—we know how to solve that, in several different ways at different price points. Because we forgot to check a null pointer? Easily solvable. And so on.

If our profession is ever to take its rightful place as a pillar of society, we need to adopt this idea from another pillar.

What do you think are the never events in software, and how should we prevent them? Use the link at the end of my bio to tell me. As always, readers will be identified only by first names, unless they request otherwise. ■

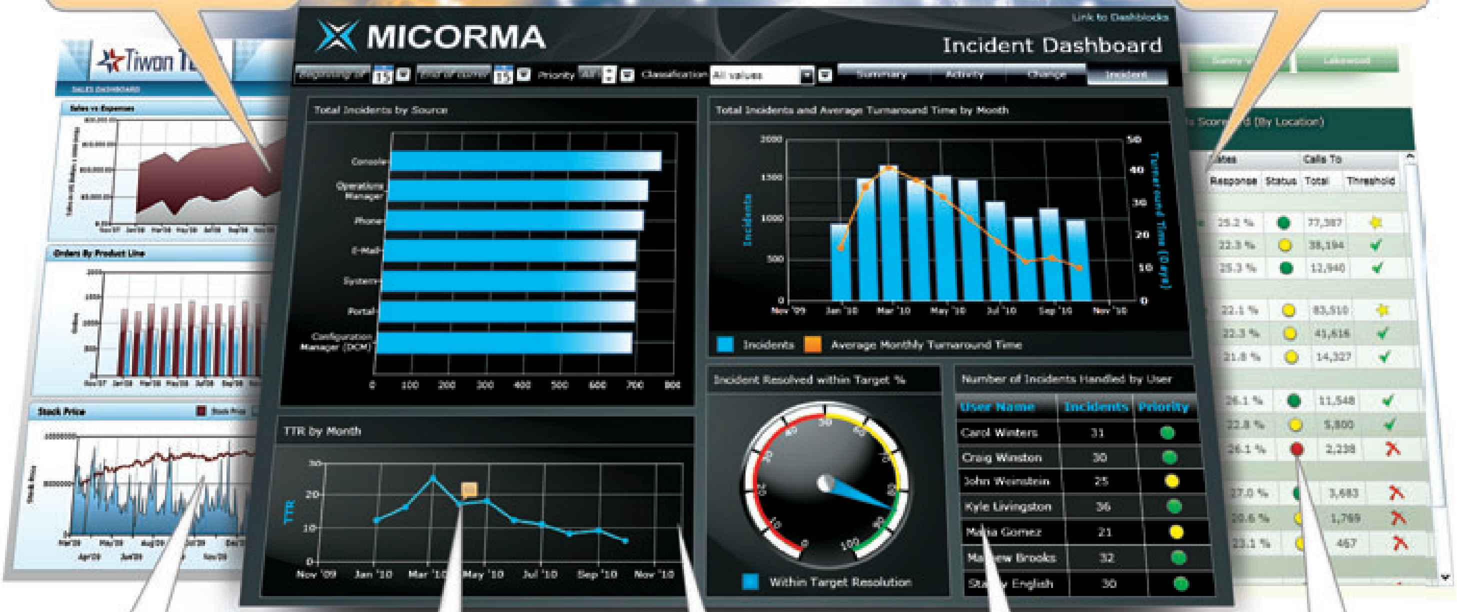
DAVID S. PLATT *teaches Programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

Dundas Dashboard

The Dashboard Framework for Developers like you

Support For Numerous Data Sources

Extensible And Customizable With Our Open API



User Name	Incidents	Priority
Carol Winters	31	●
Craig Winston	30	●
John Weinstein	25	●
Kyle Livingston	36	●
Ma ia Gomez	21	●
Ma ew Brooks	32	●
St y English	30	●

Rapid Dashboard Development

Dundas Communication (Annotations)

Full scripting Capabilities

Extend The MS BI Stack And SharePoint

Notifications/Alerts

V2.5 Now Available

Dundas Dashboard was built with developers and IT staff in mind. Whether it's our open API, simple web integration or powerful scripting capabilities, technologists have all the tools and options they need for getting their dashboard projects up and running quickly and easily.

Powered by
Microsoft® Silverlight®

Dundas
Data Visualization

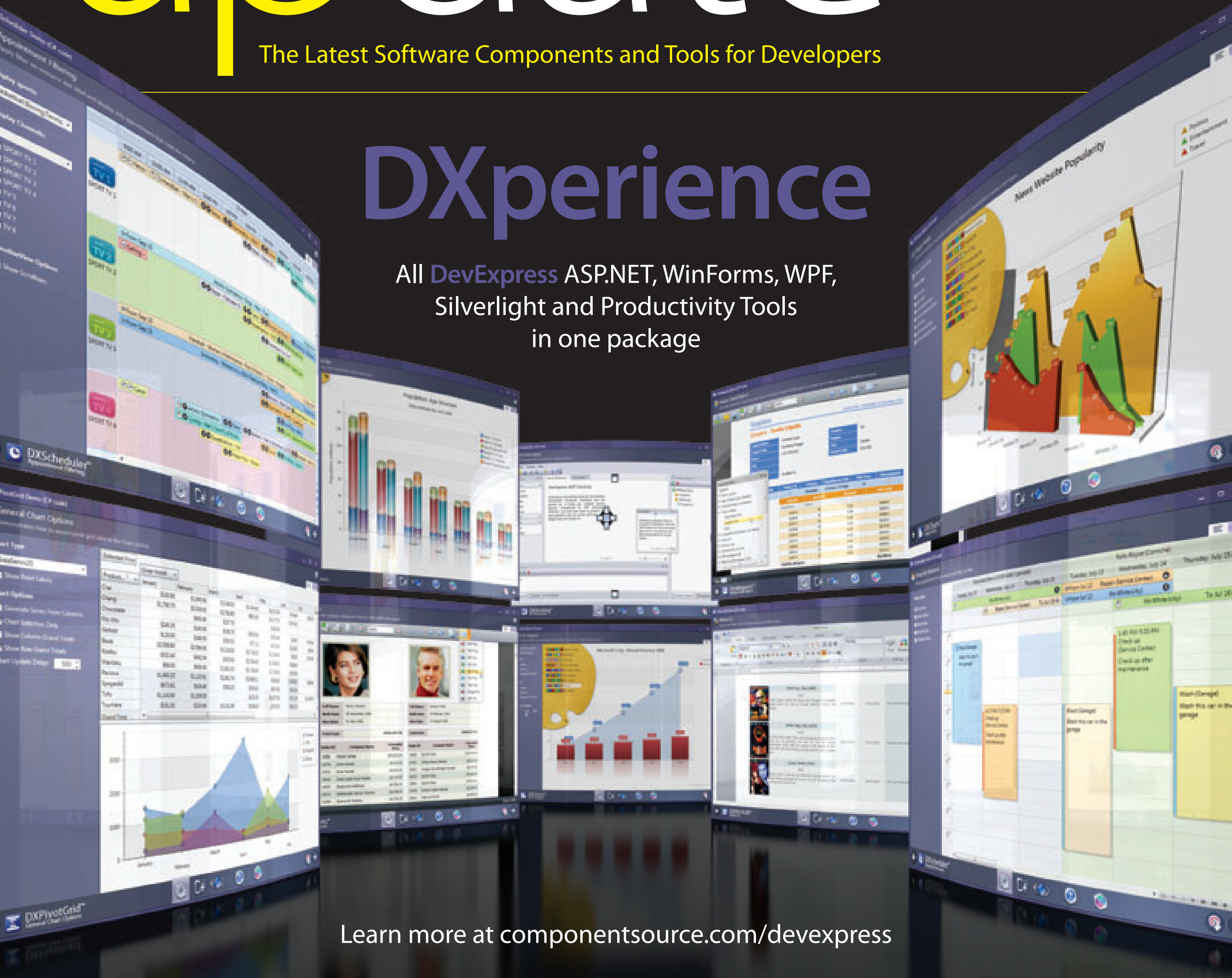
www.dundas.com
(416) 467-5100 • (800) 463-1492

/issue 77/us edition

ComponentSource®
The Definitive Source of Software Components

update

The Latest Software Components and Tools for Developers



DXperience

All DevExpress ASP.NET, WinForms, WPF, Silverlight and Productivity Tools in one package

Learn more at componentsource.com/devexpress

US Headquarters
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
USA
Tel: +1 (770) 250 6100
Fax: +1 (770) 250 6199

European Headquarters
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
United Kingdom
Tel: 0118 958 1111
Fax: 0118 958 9999

Asia / Pacific Headquarters
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japan
102-0083
Tel: +81-3-3237-0281
Fax: +81-3-3237-0282

www.componentsource.com

Sales Hotline:
(888) 850-9911

Why buy from ComponentSource?

ComponentSource offers a unique global service, used by over **1,000,000** developers worldwide.

Open 24 hours a day, Monday to Friday

International customer service centers in the US, UK and Japan.

Americas ●

ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
USA

Tel: 770 250 6100
Fax: 770 250 6199

Speakers available:
English / Spanish

Opening Hours:
9:00 am - 7:00 pm EST

EMEA ●

ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
United Kingdom

Tel: 0118 958 1111
Fax: 0118 958 9999

Speakers available:
English / French / German / Spanish / Italian

Opening Hours:
9:00 am - 5:30 pm GMT & CET

Asia-Pacific ●

ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japan
102-0083

Tel: 03-3237-0281
Fax: 03-3237-0282

Speakers available:
English / Japanese / Korean / Mandarin / Cantonese

Opening Hours:
10:00 am - 6:00 pm JST

Toll Free telephone support in 47 countries

Available in English, French, German, Italian, Spanish, Japanese, Korean, Cantonese and Mandarin.

Argentina	0800 666 3484	France	0800 90 92 62	Peru	0800 53288
Australia	1 800 0 15292	Germany	0800 186 07 06	Poland	00800 442 1092
Austria	0800 281 750	Greece	00800 44121891	Portugal	800 844 125
Belgium	0800 7 5603	Hong Kong	800 908 581	Russia	810 800 2251 1044
Brazil	0800 891 6478	Hungary	06800 16674	Singapore	800 810 2136
Bulgaria	00800 115 4445	India	000 800 44 01 455	Slovenia	0800 80297
Canada	888 850 9911	Ireland, Rep.	1 800 535 661	South Africa	0800 99 1088
Chile	1230 020 6857	Israel	180 924 2003	Spain	900 93 8926
China (North)	10800 481 1661	Italy	800 790046	Sweden	020 794 989
China (South)	10800 813 1594	Japan	0120 343 550	Switzerland	0800 83 5305
Colombia	01800 710 2043	Korea, Republic of	00798 14 800 6332	Taiwan	00 801 814313
Croatia	0800 222550	Mexico	01 800 681 1559	Turkey	00 800 4491 3617
Czech Republic	800 143 313	Netherlands, The	0800 022 8832	United Kingdom	0800 581111
Denmark	80 88 17 20	New Zealand	800 449181	United States	888 850 9911
Ecuador	1 800 010 562	Norway	800 1 3685	Uruguay	000 401 902 38
Finland	0800 1 18002	Panama	00 1 800 203 1587	Venezuela	0 800 100 9103

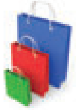
Localized Web

Browse our multilingual sites or use our Live Chat service.

Pay via purchase order, check, bank transfer or credit card in 20 currencies.

Product Choice

Choose from over 10,000 hand-picked, commercial grade Product SKU's.



Benefit from Back-2-Back Support™, our unique service level guarantee.

Impartial Advice

Get free, independent advice from our product experts.

With over 15 years experience, we know the market.

Trusted Supplier

As Official Distributors, we also supply Corporate, Reseller and Government institutions globally.

Single Source

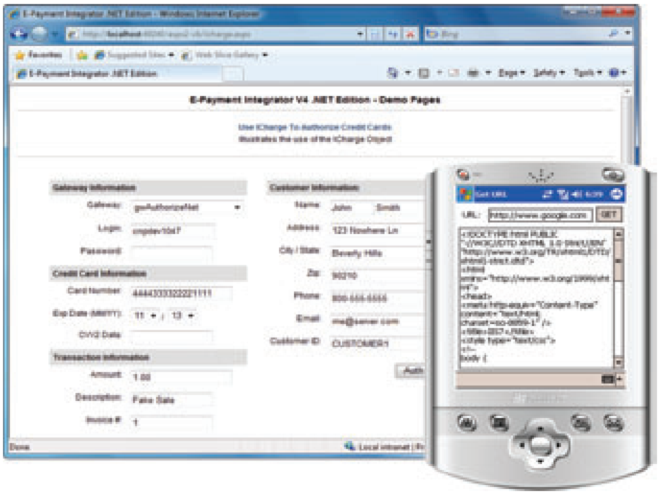
For all your Software Projects, we are your one-stop shop.

/n software

/n software Red Carpet Subscription Q1 2011

Write communications, security and e-business applications.

- Components for FTP, IMAP, SNMP, SSL, SSH, S/MIME, digital certificates, credit card processing and e-business (EDI) transactions
- Integrate SharePoint Site, List, and Document management functions



E-Payment Integrator V4 .NET Edition - Demo Pages

Use ICharge To Authorize Credit Cards
Illustrates the use of the ICharge Object

Gateway Information
Gateway: gwAuthorizeNet
Login: cnpdev1047
Password:

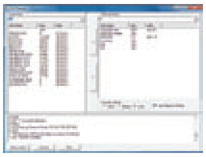
Credit Card Information
Card Number: 4444333322221111
Exp Date (MMYY): 11 / 13
CVV2 Data:

Transaction Information
Amount: 1.00
Description: Fake Sale
Invoice #: 1

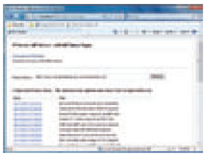
Customer Information
Name: John Smith
Address: 123 Nowhere Ln
City / State: Beverly Hills
Zip: 90210
Phone: 800-555-5555
Email: me@server.com
Customer ID: CUSTOMER1

URL: http://www.google.com GET

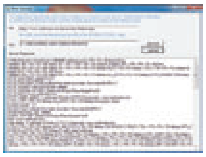
/n software Red Carpet Subscription



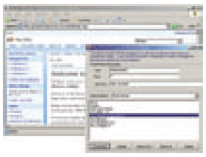
IP*Works!
A comprehensive framework for Internet development.



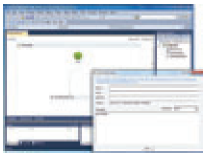
InternetC ommunications
ATOM, REST, MX, DNS, RSS, NNTP, SMPP, POP, Rexec, Rshell, SMTP, SOAP, Telnet, Ping, TFTP, FileMailer, NetClock, FTP, HTTP and more.



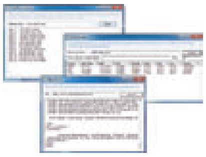
SecureC omponents
/n software Red Carpet Subscription includes secure versions of many of the IP*Works! components.



SharePoint Integrator Products
Interface directly with SharePoint Server and provide access to a wide range of SharePoint data access and management capabilities.



Windows WorkflowF oundation Activities
Includes activities that make it easy to integrate Internet capabilities into your workflow packages.



Everything in One Subscription
There are no technology restrictions, you get everything, .NET, COM, C++, Java, UNIX, Linux even Mac OS X and iPhone.

One developer, multiple workstations, all /n software products.

/n software Red Carpet Subscription gives you components for every major protocol from FTP to IMAP to SNMP, SSL and SSH security, S/MIME encryption, digital certificates, credit card processing, ZIP compression, shipping & tracking, e-business (EDI) transactions and robust Enterprise ready Internet connectivity, secure messaging and file transfer for Windows Workflow Foundation. There are no licensing restrictions, so you can code on your work machine, your notebook or even your home machine. Plus there are no technology restrictions, so you get everything: .NET, COM, C++, Java, Pocket PC, UNIX, Linux, Mac OS X and iPhone. You receive quarterly update CDs with free updates, upgrades and new releases. It includes more than $30,000 worth of software including: all /n software products - IP*Works!, SSL, S/MIME, S/Shell, Zip, EDI, Amazon Integrators, Shipping Integrators, SharePoint Integrators, Integrators for QuickBooks, Enterprise Adapters, Paymentech, E-Payment, Vital/TSYS, USPS, PaytPal, Workflow Activities etc.

/n software Red Carpet Subscription includes IP*Works! which is a comprehensive framework for Internet development and is the core building block for most /n software products. IP*Works! eliminates the complexity of Internet development by providing easy-to-use, programmable components that facilitate tasks such as sending email, transferring files, managing networks, browsing the Web, consuming Web services, etc.

It includes support for all major internet protocols: ATOM, DNS, FileMailer, FTP, HTMLMailer, HTTP, ICMPPort, IMAP, IPDaemon, IPInfo, IPMonitor, IPPort, LDAP, MCast, MIME, MX, NetClock, NetCode, NetDial, NNTP, Ping, POP, RCP, REST, Rexec, Rshell, RSS, SMPP, SMTP, SNPP, SOAP, Syslog, Telnet, TFTP, TraceRoute, UDPPort, WebDav, WebForm, WebUpload, Whois, XMLp and XMPP.

Also included in the /n software Red Carpet Subscription is IP*Works! SSL which is a suite of secure, SSL-enabled drop-in replacements for the "classic" IP*Works! components.

The components are fast, robust, and reliable and consume a minimum of resources. They are small and lightweight with no dependencies on external libraries. Included are native development components for all supported platforms. IP*Works! includes a complete unified framework with a common, easy to learn object model and simplified interfaces allowing you to be more productive. The components are rigorously tested having undergone thousands of hours of testing, you also get reference documentation, hundreds of sample applications and fully indexed help files.

FROM /N SOFTWARE

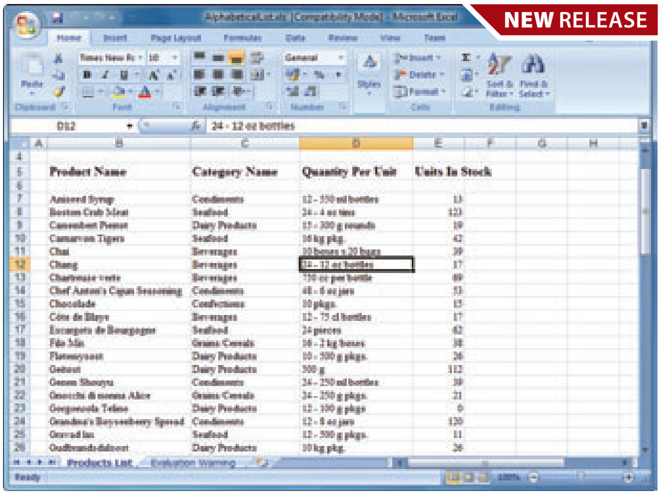
Product	Top 250	Review	Price
IP*Works!E DI .NETE dition V8.0	98	-	$489.02
/n software Red Carpet Subscription Q1 2011	100	-	$1,424.05
IP*Works! ActiveXC OME dition V8.0	180	-	$629.10
IP*Works!SSL C++E dition V8.0	204	-	$949.05
IP*Works!Z ip .NETE dition V8.0	212	★★★	$369.08
IP*Works! C++E dition V8.0	222	-	$759.05
IP*Works!SSL PHPE dition V8.0	228	-	$809.10
IP*Works!SSL .NETE dition V8.0	231	-	$809.10
QuickBooksI ntegrator .NETE dition V5.0	-	-	$539.10
IP*Works!Z ip JavaE dition V8.0	-	-	$449.10
IP*Works! .NETE dition V8.0	-	-	$539.10
/ns oftwareN etCmdlets V2.0	-	-	$77.42
/ns oftwareShar ePointE xtensions V3.0	-	-	$587.02
/ns oftware WorkflowA ctivities V3	-	-	$97.02
AmazonI ntegrator .NET V4	-	-	$587.02
E-BankingI ntegrator JavaE dition V3	-	-	$539.10
E-PaymentI ntegrator ActiveXE dition V5	-	-	$379.05
PaymentechI ntegrator .NET Edition V6.0	-	-	$2,449.02
PayPalI ntegrator JavaE dition V3.0	-	-	$391.02
PayPalI ntegrator ActiveX/ VB Edition V3.0	-	-	$391.02

Aspose

Aspose.Total .NET Q1 2011

Create, manage and convert many popular document formats.

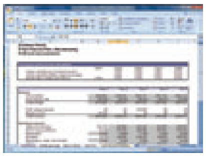
- Supports Word, Excel, PowerPoint, Project, PDF and Flash file formats
- Demos written in C# and VB.NET highlight the best practices for implementing many common tasks



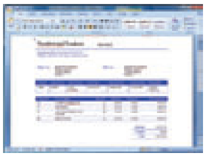
D12 | 24 - 12 oz bottles

Product Name	Category Name	Quantity Per Unit	Units In Stock
Aniseed Syrup	Condiments	12 - 550 ml bottles	13
Boston Crab Meat	Seafood	24 - 4 oz tins	123
Camembert Pierrot	Dairy Products	15 - 300 g rounds	19
Carnarvon Tigers	Seafood	16 kg pkg.	42
Chai	Beverages	10 boxes x 20 bags	39
Chang	Beverages	24 - 12 oz bottles	17
Chartreuse verte	Beverages	750 cc per bottle	69
Chef Anton's Cajun Seasoning	Condiments	48 - 6 oz jars	53
Chocolade	Confections	10 pkgs.	15
Côte de Blaye	Beverages	12 - 75 cl bottles	17
Escargots de Bourgogne	Seafood	24 pieces	62
Filo Mix	Grains/Cereals	16 - 2 kg boxes	38
Flotemysost	Dairy Products	10 - 500 g pkgs.	26
Geitost	Dairy Products	500 g	112
Genen Shouyu	Condiments	24 - 250 ml bottles	39
Gnocchi di nonna Alice	Grains/Cereals	24 - 250 g pkgs.	21
Gorgonzola Telino	Dairy Products	12 - 100 g pkgs	0
Grandma's Boysenberry Spread	Condiments	12 - 8 oz jars	120
Gravad lax	Seafood	12 - 500 g pkgs.	11
Gudbrandsdalsost	Dairy Products	10 kg pkg.	26

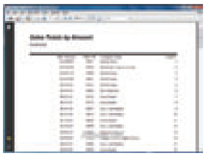
Aspose.Total



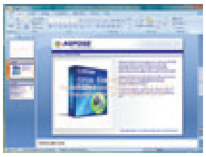
Aspose.Cells
Lets you create and manage Excel Spreadsheets without requiring Microsoft Excel.



Aspose.Words
Lets you read, modify and write Word documents without utilizing Microsoft Word.

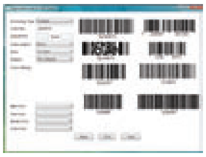


Aspose.Pdf
Lets you create PDF documents without utilizing Adobe Acrobat.



Aspose.Slides
Enables your application to read and write PowerPoint documents without utilizing Microsoft PowerPoint.

Aspose.Pdf.Kit
Easily manipulation existing PDF documents.



Aspose.BarCode
Barcode generation and recognition component, written in managed C# for your WinForms, ASP.NET and .NET CF applications.

Aspose.Total .NET offers every .NET component that Aspose has available as well as all new components released during the annual subscription.

Aspose.Total for.NET offers every .NET component that Aspose has available. With it you will be able to programmatically manage some of the most popular file formats used in business: Word, Excel, PowerPoint, Project, PDF and Flash. Aspose Total .NET also supports mail merge, creating shapes on slides, extracting text from PDF files, creating adhoc reports with graphs and pivot tables and more. Your Aspose Total .NET purchase includes 1 year of maintenance and support so you will obtain all updates and new Aspose Total .NET components released during the year.

Aspose.Words for .NET is a class library that enables you to perform a wide range of document processing tasks directly within your .NET applications. Aspose.Words for .NET supports DOC, OOXML, RTF, HTML, OpenDocument, PDF, XPS, EPUB and many other formats. With Aspose.Words you can generate, modify, convert, render and print documents without using Microsoft Word.

Aspose.Cells for .NET provides a flexible group of components that enable .NET applications to create and manage Excel Spreadsheets without requiring Microsoft Excel to be installed on the server.

Aspose.Pdf for .NET is a PDF document creation component that enables your .NET applications to read, write and manipulate PDF documents without using Adobe Acrobat.

Aspose.Slides for .NET is a PowerPoint management component that enables .NET applications to read, write and manipulate PowerPoint documents without using Microsoft PowerPoint. Aspose.Slides for .NET supports formats including PPT, PPS, POT and PresentationML (OOXML, PPTX).

Aspose.BarCode for .NET is a robust and reliable barcode generation and recognition component.

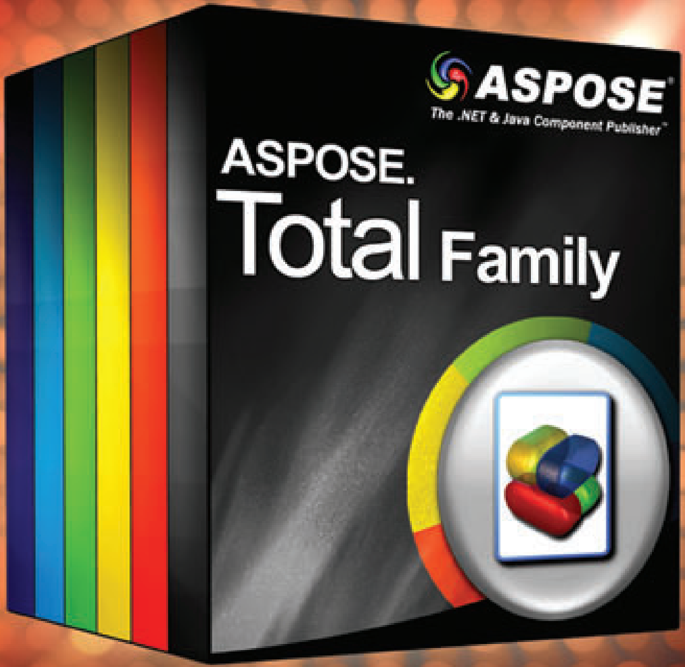
Aspose.Tasks for .NET enables .NET applications read and write Microsoft Project documents without utilizing Microsoft Project.

Aspose.Network for .NET is a suite of components for professional network programming, which includes implementations of most of the major network protocols.

Aspose.Report for .NET can be used for a variety of tasks, ranging from creating adhoc reports to building quick and easy search screens and production of a variety of great looking charts.

FROM ASPOSE

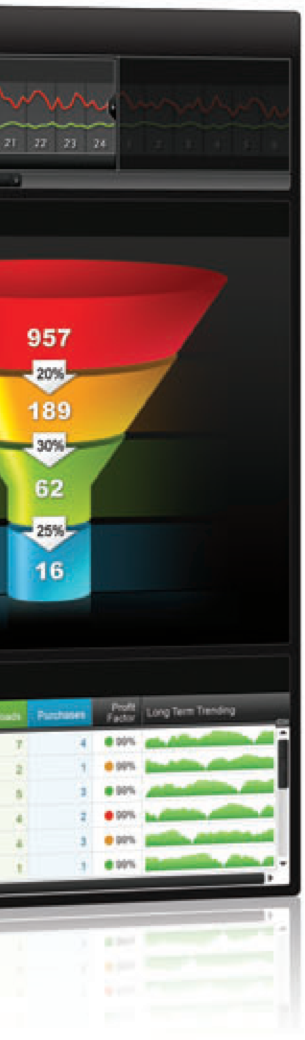
Product	Top 250	Review	Price	Product	Top 250	Review	Price
Aspose.Total .NET Q12011	18	-	$1,959.02	Aspose.BarCodef orJas perReports V1.2	-	-	$391.02
Aspose.Cells Java V2.4	25	-	$881.02	Aspose.BarCodef orRe portingS ervices V2.2	-	-	$391.02
Aspose.Words .NET V9.3	31	-	$881.02	Aspose.BarCodeP roductF amily	-	-	$636.02
Aspose.Words Java V3.3	72	-	$881.02	Aspose.Cellsf or ReportingS ervices V1.5	-	-	$881.02
Aspose.Slides .NET V4.2	117	-	$587.02	Aspose.Cellsf or SharePoint V1.0	-	-	$881.02
Aspose.BarCode Java V2.0	158	-	$391.02	Aspose.CellsP roductF amily	-	-	$1,469.02
Aspose.Cells .NET V5.1	202	-	$881.02	Aspose.Flash .NET V2.9	-	-	$391.02
Aspose.Pdf .NET V4.6	-	-	$587.02	Aspose.Flash for Reporting Services (SSRS) V1.1.0	-	-	$391.02
Aspose.Tasks .NET V3.0	-	-	$587.02	Aspose.FlashP roductF amily	-	-	$636.02
Aspose.BarCode .NET V3.5	-	★★★★★	$391.02	Aspose.MetaFiles Java V1.6.1	-	-	$195.02


ASPOSE®
The .NET & Java Component Publisher™
ASPOSE.
Total Family

Presets: ○ Day ○ Week ○ Month ○ Quarter ● Year

Date	Country	Product
Wed Sep 1 2009 2:09:35 AM	US	Product A
Wed Sep 1 2009 6:03:14 AM	US	Product B
Wed Sep 1 2009 9:31:23 AM	Canada	Product C
Wed Sep 1 2009 10:56:54 AM	US	Product C
Wed Sep 1 2009 11:51:18 AM	US	Product C
Wed Sep 1 2009 1:20:20 PM	US	Product B
Wed Sep 1 2009 2:29:44 PM	France	Product A

Microsoft Silverlight   Microsoft Visual Studio



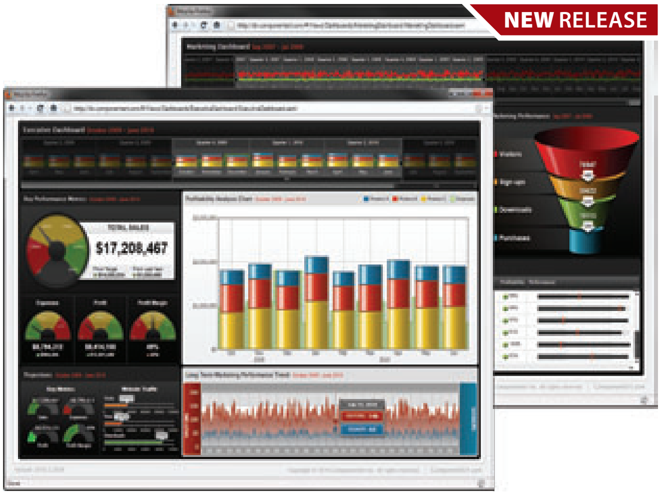
957
20%
189
30%
62
25%
16
Purchases
Profit Factor
Long Term Trending

ComponentArt

ComponentArt Data Visualization for .NET Ultimate 2011

Controls for digital dashboards and data analysis applications in Silverlight and WPF.

- Enterprise-grade charting, gauges and maps along with state-of-the-art data display and processing components
- Brilliant rendering, interactive drill-downs, zooming & scrolling



ComponentArt Data Visualization for .NET Ultimate

A Highly Interactive Control for Date Range Selections
TimeNavigator was designed to enable quick and intuitive date range selections.

Gauges
A wide selection of digital gauge controls, featuring easy KPI binding, customizable pointers, scales & ranges and smooth animation.

Bar & Column Charts
includes a variety of attractive 2D and 3D bar and column chart shapes.

Charts
Includes a variety of 2D and 3D shapes and the ability to customize data point annotations and popups.

Effective Spreadsheet and Paged Display of Data
Choose between two interfaces for tabular data: a spreadsheet and a traditional paged data grid.

Effective Visualization of Geographic Data
Included is a large collection of over 150 ready-to-use maps of continents, regions and countries.

A complete feature set for building next-generation digital dashboards and data analysis applications.

ComponentArt Data Visualization for .NET Ultimate is a suite of Silverlight and WPF controls for the development of digital dashboards and data analysis applications. It includes charting, gauges and maps along with data display and processing components.

Charting: A set of charting components (XY, pie, funnel, radar, tree map, legend, drill down manager, WCF data provider, etc) designed to deliver interactive visualization of business data. The components feature brilliant rendering, drill-downs, built-in zooming & scrolling and annotation objects. Enterprise charting features include the ability to handle very large datasets and CalcEngine integration.

Gauges: A wide selection of digital gauge controls (radial linear, numeric, bullet graph, cylinder, half donut, thermometer, etc). The components feature easy KPI (key performance indicator) binding, customizable pointers, scales and ranges, flexible inner layout and smooth animation. All controls include professional skins and themes.

Maps: Dynamic, data-driven maps designed to provide visualization of geographic data. The components feature drill-downs, multiple shape layers, tooltip & point templates and flexible projections. Includes over 100 maps covering all world continents & regions, major countries, all US states and custom maps.

GridView: A data grid control which delivers a visually rich display of tabular data. The components feature data grouping, sorting, filtering, and operating in "spreadsheet" or "paged grid" modes. Enterprise grid features include handling very large datasets, cell formulas, calculated styles and CalcEngine integration.

TimeNavigator: An innovative control that allows quick, interactive and intuitive date range selections. It features hierarchical date and time drilldowns: decade, year, quarter, month, week, day, hour, minute, second. The component also includes customizable fiscal year offset, list of business days and/or holidays, international calendars and more.

CalcEngine: A general purpose data processing engine, used to perform complex operations on a given dataset. It features ComponentArt "Calc", an interpreted expression language for dynamic data transformations on the client. The CalcEngine is loaded with useful & functional libraries including: set operations, statistical & financial functions.

Subscription License includes all minor and major product updates for a period of one year.

FROM COMPONENTART

Product	Top 250	Review	Price
ComponentArt Web.UI for ASP.NET AJAX 2011	76	★★★★★	$783.02
ComponentArt UI Framework for .NET 2011	82	★★★★★	$1,077.02
ComponentArt Data Visualization for .NET Ultimate 2011	-	-	$5,879.02
ComponentArt Charting for Silverlight 2011	-	-	$881.02
ComponentArtC hartingf or WPF 2011	-	-	$881.02
ComponentArt Data Visualization for Silverlight 2011	-	-	$1,959.02
ComponentArtD ata Visualizationf or WPF 2011	-	-	$1,959.02
ComponentArt Gauges for Silverlight 2011	-	-	$587.02
ComponentArtG augesf or WPF 2011	-	-	$587.02
ComponentArt GridView for Silverlight 2011	-	-	$587.02
ComponentArtG ridViewf or WPF 2011	-	-	$587.02
ComponentArt Maps for Silverlight 2011	-	-	$587.02
ComponentArtMap sf or WPF 2011	-	-	$587.02
ComponentArt SOA.UI for .NET 2011	-	★★★★★	$587.02
ComponentArt TimeNavigatorf or Silverlight 2011	-	-	$587.02
ComponentArt TimeNavigatorf or WPF 2011	-	-	$587.02
ComponentArt Web.UIf orS ilverlight 2011	-	★★★★	$783.02
ComponentArt Win.UIf or WPF 2011	-	★★★★★	$783.02

ComponentOne

ComponentOne Studio Enterprise 2011 v1

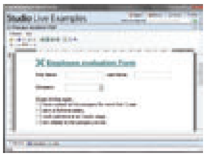
Components for Silverlight, WPF, .NET, ASP.NET, ActiveX & Mobile Devices.

- Full suite of rich and flexible .NET Windows Forms components
- Rapid development of innovative & demanding Web applications
- Includes innovative Silverlight controls

ComponentOne Studio Enterprise

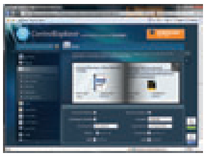
MicrosoftO utlook-style Scheduler
Schedule one-time, all day, or recurring appointments with reminders, labels, etc.

Advanced Charting for your WPF Applications
Brings you powerful rendering, animations and data-binding capabilities.

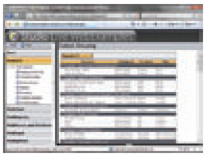


ComponentOne PDF for WinForms
Supports AcroForms and AcroFields.

iPhone and iTouch
Includes ASP.NET components developed to mimic the look and feel of the native UI of the iPhone and iTouch.



ComponentOne Book for Silverlight
Present UIElement objects as if they were pages in a regular paper book.



Create Responsive Grid Applications with Built-in AJAX
Easily group, sort, filter, resize, and reorder columns at run time, all through simple drag-and-drop.

Create rich user interfaces for Silverlight, WPF, .NET, ASP.NET, ActiveX, Mobile and Windows Forms applications.

ComponentOne Studio Enterprise includes hundreds of components for Silverlight, WPF, .NET, ASP.NET, ActiveX, iPhone & mobile devices. It offers a complete solution that targets all runtime environments and all application layers (data, presentation, reporting and transformation).

Silverlight toolset includes: Feature packed Silverlight components (grid, chart, menu, treeview, uploader, layout, data input, gauges, maps, image rotator, navigation, etc). You can use expression blend to fully style the Silverlight controls which allows developers to easily create rich graphical interfaces for their Rich Internet Applications. Built with XAML, the Silverlight controls promote collaboration in your team and fit seamlessly into your developer & designer workflow. Furthermore, you can take XAML from Silverlight and use it in your WPF application. Also available from ComponentOne is the XAP Optimizer which helps you optimize the size of the XAP files within your Silverlight applications.

WPF toolset includes: Schedule for WPF - a suite of tools delivering scheduling, month calendar and multi-month calendar components for the WPF platform. Chart for WPF - visually stunning 2D/3D bar, area, line, pie and column charts. Grid for WPF - a set of controls that allow you to integrate grids into your WPF applications. Reports for WPF - includes a set of WPF reporting tools: the C1Report, C1PrintDocument, and C1DocumentViewer components as well as the C1ReportDesigner stand-alone application.

ASP.NET toolset includes: Styled, easy-to-use controls built on Web standards including AJAX, CSS, and XHTML. Components include: eCommerce (PayPal), file format management (PDF, Flash, XLS, ZIP), calendars, charting (2D and 3D), rich text editing, grids, input and mask controls, menus and toolbars, reporting, scheduling and spelling features. You can also recreate the look & feel of iPhone and iTouch user interfaces using Button, LaunchPad, Calendar, NavigationList & Slider components.

WinForms toolset includes: .NET controls featuring two award-winning grids: FlexGrid and True DBGrid. FlexGrid is designed for unbound applications. It is a flexible grid control for creating user-friendly interfaces that display, edit, format, organize, summarize, and print tabular data. The strength of True DBGrid is in its data binding; with an ADO.NET managed database interface, it offers features such as Excel-like split views and built-in hierarchical binding and grouping.

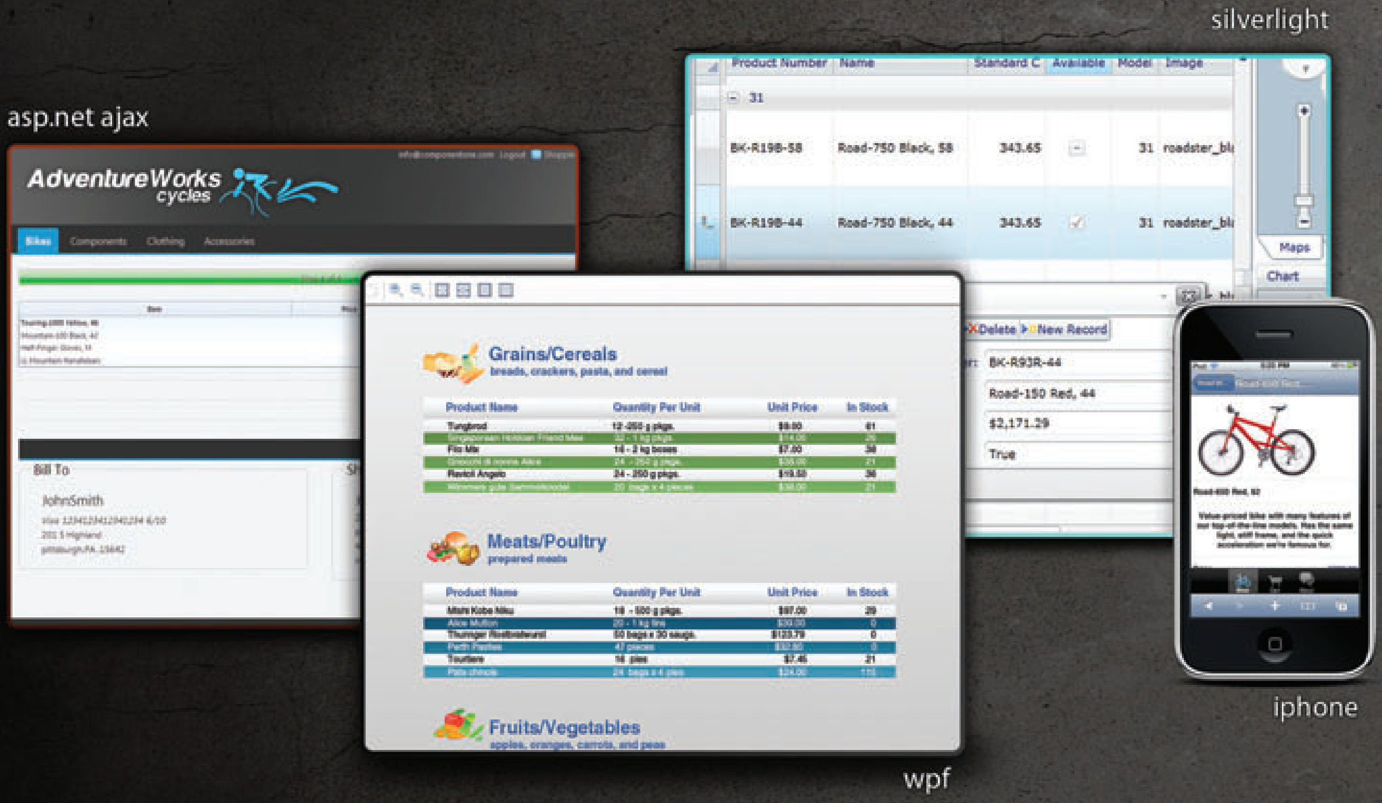
FROM COMPONENTONE

Product	Top 250	Review	Price
ComponentOne Studio Enterprise 2011 v1	8	★★★★	$1,170.00
ComponentOne Studio for WinForms 2011 v1	20	★★★	$720.00
ComponentOne Studio for Silverlight 2011 v1	69	★★★★★	$720.00
TrueD BGridP ro 8.0	77	★★★★	$490.00
ComponentOneD oc-To-Help Enterprise 2011v 1	89	-	$1,056.00
ComponentOne Studio for ActiveX 2011 v1	123	-	$720.00
ComponentOneLi veLinq 3.5	146	-	$910.00
ComponentOneD oc-To-Help For Word 2011v 1	186	-	$816.00
ComponentOneC hart 8.0	207	★★★	$490.00
VSFlexGridP ro 8.0	223	★★★★	$490.00

Product	Top 250	Review	Price
ComponentOne Chart for Winforms 2011 v1	226	-	$490.00
ComponentOne Studio for Mobile Devices 2011 v1	243	-	$720.00
ComponentOne OLAP for WinForms 2011 v1	-	-	$1,620.00
ComponentOne FlexGrid for Winforms 2011 v1	-	-	$490.00
ComponentOne Studio for ASP.NET AJAX 2011 v1	-	-	$720.00
ComponentOne Chart for Sharepoint 2011	-	-	$480.00
ComponentOne DataGrid for SharePoint 2011	-	-	$480.00
ComponentOneD emoWorks	-	-	$1,056.00
ComponentOne Editor for WinForms 2011 v1	-	-	$490.00
ComponentOneI ntelliSpell	-	-	$182.00

SERIOUSLY FUN CODING

GRIDS • CHARTS • GAUGES • REPORTS • SCHEDULES • MENUS • TOOLBARS • RIBBON • EDITORS • DATA INPUT • PDF



Product Number	Name	Standard C	Available	Model	Image
− 31					
BK-R19B-58	Road-750 Black, 58	343.65	−	31	roadster_bl
BK-R19B-44	Road-750 Black, 44	343.65	✓	31	roadster_bl

BK-R93R-44

Road-150 Red, 44

$2,171.29

True

Bill To

JohnSmith

Grains/Cereals

breads, crackers, pasta, and cereal

Product Name	Quantity Per Unit	Unit Price	In Stock
Tungbrod	12 -250 g pkgs.	$9.00	61
Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	$14.00	26
Filo Mix	16 - 2 kg boxes	$7.00	38
Gnocchi di nonna Alice	24 - 250 g pkgs.	$38.00	21
Ravioli Angelo	24 - 250 g pkgs.	$19.50	36
Wimmers gute Semmelknödel	20 bags x 4 pieces	[illegible]	21

Meats/Poultry

prepared meats

Product Name	Quantity Per Unit	Unit Price	In Stock
Mishi Kobe Niku	18 - 500 g pkgs.	$97.00	29
Alice Mutton	20 - 1 kg tins	$39.00	0
Thuringer Rostbratwurst	50 bags x 30 saugs.	$123.79	0
Perth Pasties	48 pieces	$32.80	0
Tourtiere	16 pies	$7.45	21
Pate chinois	24 boxes x 2 pies	$24.00	115

Fruits/Vegetables

apples, oranges, carrots, and peas

Road-650 Red, 62

Value-priced bike with many features of our top-of-the-line models. Has the same light, stiff frame, and the quick acceleration we're famous for.

WINFORMS WPF ASP.NET AJAX SILVERLIGHT iPHONE MOBILE ACTIVEX

Seven platforms, hundreds of controls, one studio. Build awesome desktop, Web, and mobile apps with our tools for WinForms, WPF, ASP.NET AJAX, Silverlight, iPhone, Mobile, and ActiveX. Studio Enterprise gives you the technology you need to reach any user.

ComponentOne® Studio® Enterprise 2010 v3

ComponentOne®

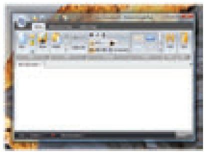
DevExpress (Developer Express)

DXperience v2010 vol 2

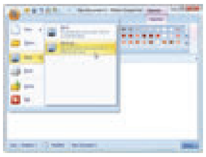
All Developer Express ASP.NET, WinForms, WPF & Silverlight components & tools.

- User Interface functionality for WinForms, ASP.NET, WPF & Silverlight
- Display digital dashboards in your Web based applications
- Includes a scheduling/planning/calendar suite for ASP.NET

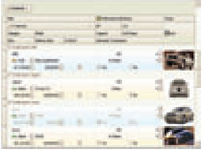
DXperience



XtraBars Suite Ribbon Control and Tabbed MDI interface
With the XtraBars Suite you can easily build Ribbons from scratch.



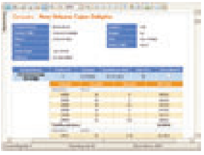
XtraBarsS uite
You can create application menus, galleries and contextual tabs in ribbons.



DXGridf or WPF
Supports both tabular and card display formats and provides facilities for the end-user to sort and group data.

AgRichEdit for Silverlight
Allows you to introduce Microsoft Word like text editing features into your next Silverlight application with ease.

XtraReports End-User Designer
The XtraReports Suite features a fully functional End-User Designer.



XtraReports Master-Detail Report
With the XtraReports Suite, you can easily build banded reports and even Master-Detail reports.

All the .NET component suites, libraries and IDE tools produced by Developer Express in one package.

DXperience includes all Developer Express products engineered for Visual Studio and the .NET Framework. Priority technical support is also included with updates and upgrades for 1 year. With this subscription, you receive key technologies by Developer Express including IDE productivity tools, business application frameworks, and components for Windows Forms, ASP.NET, WPF and Silverlight. In addition to the individual tools, you receive complete source code for the Windows Forms & ASP.NET components. With the Universal subscription, you also receive eXpressApp Framework which is an instrument for creating business applications quickly and easily.

WPF products include: DXGrid for WPF - fast server-side data processing and complete support for templates and styles, **DXCharts for WPF** - clear & compact XAML syntax and element customization via templates, **DXCarousel for WPF** - stunning navigation and data browsing, **DXNavBar for WPF** - easy and intuitive navigation.

Silverlight products include: **AgDataGrid** - fast data manipulations and rich end user experiences, **AgRichEdit** - Word like text editing features, **AgMenu** - static menus, toolbars and context menus etc, **AgLayoutControl** - layout management for Silverlight.

Windows Forms products include: **XtraGrid Suite** - a grid control with four data layout options from a simple table to banded multi-line row layout and card views, **XtraScheduler Suite** - enables you to incorporate the UI found in Office Calendar within your applications, **XtraPivotGrid Suite** - a data analysis, mining and visual reporting solution, **XtraGauges** - linear, circular, digital and state indicator gauge types, **XtraEditors** a library of over 20 editors that can be used both standalone or embedded, **XtraWizard** - easily generate multi-step wizard dialogs that fully conform to Wizard 97 or Aero standards.

ASP.NET products include: **ASPxGridView and Editors Library** - a robust grid and data editors library for ASP.NET, **ASPxPivotGrid Suite** - a multi-dimensional analysis component, **ASPxNavBar Suite** - Web navigation bar with full AJAX support, **ASPxMenu Suite** - an extremely lightweight, customizable main and context menu and more.

Also included in the Universal and Enterprise Editions are Visual Studio productivity add-ins: CodeRush, Refactor! Pro and eXpress Persistent Objects.

FROM DEVEXPRESS (DEVELOPER EXPRESS)

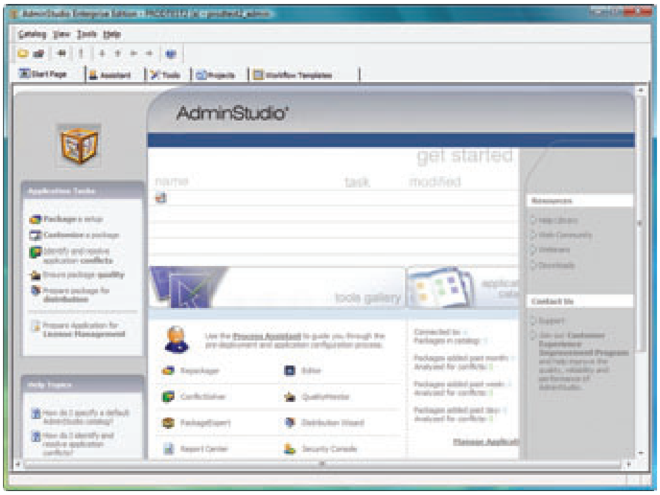
Product	Top 250	Review	Price
DXperience v2010v ol2	1	★★★★	**$1,234.99**
DXperience WinForms v2010v ol2	3	★★★★	**$711.99**
DXperience ASP.NET v2010 vol 2	15	★★★★	**$711.99**
XtraGrid .NET Suite v2010 vol 2	22	★★★★	**$284.99**
DXperience WPF v2010v ol2	49	-	**$759.99**
Developer Express VCL Subscription	83	-	**$1,469.99**
XtraReports Suite v2010 vol 2	96	★★★★★	**$332.49**
XtraPivotGrid Suite v2010 vol 2	104	-	**$284.99**
XtraPrinting Library v2010 vol 2	116	★★★★★	**$142.49**
DXperience Silverlight v2010 vol 2	128	-	**$711.99**
XtraCharts Suite v2010 vol 2	144	★★★★★	**$237.49**
ASPxGridView and Editors Suite v2010 vol 2	151	-	**$293.99**
XtraVerticalGrid v2010v ol2	169	-	**$239.99**
ExpressQuantumPack	179	★★	**$682.49**
XtraTreeList Suite v2010 vol 2	242	-	**$189.99**
ASPxPivotGrid v2010v ol2	247	★★★★	**$146.99**
ExpressQuantumGridS uite V6.53	249	★★★★★	**$341.24**
XtraBars Suite v2010 vol 2	-	★★★★	**$208.99**
ExpressPack	-	-	**$487.49**
CodeRush for Visual Studio .NET v2010 vol 2	-	-	**$237.49**

Flexera Software

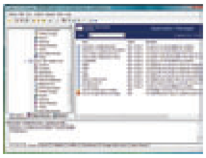
AdminStudio Enterprise V9.5

Prepare application packages for reliable enterprise deployment.

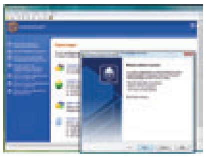
- Convert native Installshield setups into 100% pure MSI packages
- Includes tools to repackage software into Windows 7-ready MSIs
- PackageExpert tool facilitates automated QA testing



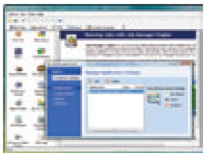
AdminStudio



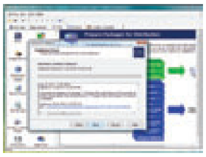
ApplicationM anager
Provides a single, user-friendly view into all the data in the Application Catalog.



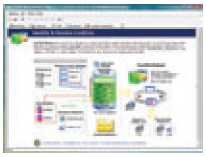
Capture Context Data During Repackaging
AdminStudio's Repackager captures context data during the MSI repackaging process.



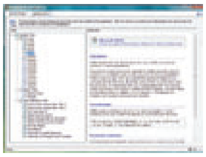
JobM anager
Schedule conflict testing tasks to run anytime without human intervention.



Hand Off Packages to Any Deployment Solution
AdminStudio makes it easy for you to release software packages to any software deployment solution.



Eliminate Software Conflicts
Automatically detect and eliminate conflicts between Windows Installer packages and installed apps before you deploy.



Diagnose and Repair Installed Applications
Determine if an installed application needs repair, what repair is needed, and why.

Quickly prepare reliable MSI and virtual packages for error-free software deployment to any Windows OS, including Windows 7.

AdminStudio lets you customize installations without having to edit Windows Installer database tables, or even write a single line of script. Developed specifically for System Administrators, Desktop Engineers and Corporate IT Project Teams, AdminStudio provides a wizard driven interface to navigate the repackaging process. Repackaging can be done with the standard default options or tailored to meet specific needs via detailed exclusion lists, the option to capture application or system-level changes, output preview, registry settings and more.

For Windows 7 projects AdminStudio's Repackager can convert older packages into MSIs that will run on Windows 7.

AdminStudio helps ensure Windows 7 application compatibility, repackages applications with the flexibility to support many deployment formats, hands off completed packages to popular deployment systems for distribution and manages a repeatable process. AdminStudio also incorporates MSI 5.0, which many organizations will use as a standard for Windows 7 migration.

AdminStudio can prepare applications for deployment on today's leading application virtualization systems: Microsoft App-V and VMware ThinApp. AdminStudio also includes easier packaging of traditionally challenging applications, such as Adobe Creative Suite 3 and 4. AdminStudio is able to create a standard process for creating virtual packages of applications that are challenging to prepare using the native Microsoft App-V Sequencer. Because AdminStudio automatically looks for all of the required information about an installation, there aren't any steps left out in preparing the application for virtual deployment.

The included Application Manager provides a single, user-friendly view into all the data and packages in the Application Catalog. It includes a summary dashboard showing high-level statistics, a list of recently modified packages and recent catalog management activity. You can also run searches on the Application Catalog from anywhere to quickly locate groups of packages, applications, merge modules, patches, and more. Packages in the catalog can be grouped using the Web-based interface to mirror your baseline systems or any user group and business unit in your environment, making it easier to test new packages before deployment and manage installed applications. You can easily create corporate validation rules that run against every application to ensure your guidelines are always followed - and no programming is needed.

AdminStudio also includeds a Process Assistant which takes all the complexity out of preparing applications for deployment. It decreases the time required for new AdminStudio users to prepare software packages for deployment, while also enabling veteran systems administrators to efficiently jumpstart a new packaging project.

FROM FLEXERA SOFTWARE

Product	Top 250	Review	Price
InstallShield Professional 2011	4	-	$1,979.01
InstallShield Premier 2011	10	-	$3,959.01
InstallAnywhere EnterpriseE dition 2010	29	-	$4,386.52
AdminStudio Professional V9.5	30	-	$5,939.01
AdminStudio Standard V9.5	45	-	$2,474.01
InstallShieldE xpress 2011	55	★★	$593.01
AdminStudio Enterprise V9.5	129	-	$8,909.01
InstallAnywhere StandardE dition 2010	130	-	$1,949.03
AdminStudiowi th VirtualizationP ack Standard V9.5	-	-	$5,444.01
AdminStudiowi th VirtualizationP ack Professional V9.5	-	-	$8,909.01
AdminStudiowi th VirtualizationP ack Enterprise V9.5	-	-	$11,879.01
InstallShield Collaboration V11.5	-	-	$395.01
InstallShield with Virtualization Pack Premier 2011	-	-	$6,929.01
InstallShield with Virtualization Pack Professional 2011	-	-	$4,949.01

THANKS for Making DevExpress #1

BESTSELLING PRODUCT AWARDS
Awards
2009-10
ComponentSource®
The Definitive Source of Software Components

#1 PRODUCT
ComponentSource® Awards 2009-10

BESTSELLING PUBLISHER AWARDS
Awards
2009-10
ComponentSource®
The Definitive Source of Software Components

#1 PUBLISHER
ComponentSource® Awards 2009-10

Shortcuts
XtraReports
Side-by-Side Reports

FLEXERA™
SOFTWARE

If Software Installations, Deployments, or Virtualization Is Your Job, Flexera Software Has Your Solution

FLEXERA SOFTWARE™
InstallShield®

InstallShield® 2011 – ***How the World Installs Software on Microsoft® Windows™***

Author reliable MSI and InstallScript installations and App-V™ virtual packages for Windows® platforms.

Optimise Installations for Windows 7
Easily configure installations that run reliably on Windows 7 and leverage Windows Installer 5 functionality.

Support the Latest Microsoft Technologies
Including Visual Studio 2010, .NET Framework 4.0, IIS 7.0, SQL Server 2008 SP1, and Windows Server 2008 R2, and App-V.

Install Software on Any Device
Extend installations to configure database servers, Web services, mobile devices, and more.

What's New in InstallShield
Includes over 400 new and enhanced features from customer requests, including a revamped IDE, superior IIS tools, virtual machine detection, Unicode support, and much more!

FLEXERA SOFTWARE™
InstallAnywhere®

InstallAnywhere™ 2010 – ***Multiplatform Installation Solution to Install Cross-Platform Applications***

Deliver a professional and consistent installation experience for Windows, Linux, Mac OS X, Solaris, AIX, HP-UX, IBM iSeries, and more.

Impress End Users with Reliable Installers
Cut your support costs and impress end users with a professional, branded installer that installs software correctly every time.

One Installation for 31 Languages
Installers can be automatically translated into 31 languages to help your applications reach end users around the world.

Perfect for Complex Desktop and Server Applications
Built-in customization tools make quick work of even your most complex installer requirements.

What's New in InstallAnywhere 2010
Breakthrough functionality for managing all phases of the installation lifecycle, including installation, rollback, maintenance, repair, and uninstall. Also expanded platform support for new platforms like Windows 7, Windows Server 2008 R2, SUSE Linux 11.2, Ubuntu 9.10, and Mac OS X 10.6.

FLEXERA SOFTWARE™
AdminStudio®

AdminStudio® – ***How IT Prepares Reliable MSI and Virtual Packages for Error-Free Deployment***

Prepare reliable MSI and virtual packages for deployment. It has automated tools that speed software through the packaging process.

Cut Packaging Time by 70%
Automated tools help you get applications to end users up to 70% faster.

Deploy Virtual Applications 9x Faster
AdminStudio automatically converts MSIs to Microsoft® App-V™, VMware® ThinApp™* or Citrix® XenApp™ virtual packages–9x faster.

Migrate MSIs to Windows™ 7 Automatically
AdminStudio can proactively determine which applications are good candidates for migrating to Windows 7 without a second round of repackaging.

Works with Any Deployment Solution
AdminStudio supports leading deployment tools such as Microsoft® Configuration Manager, Novell® ZENworks®, LANDesk® Management Suite, and much more.

** Requires separate purchase of VMware ThinApp*

BESTSELLING PUBLISHER AWARDS
Awards 2009-10
ComponentSource
#4 PUBLISHER
ComponentSource Awards 2009-10

BESTSELLING PRODUCT AWARDS
Awards 2009-10
ComponentSource
#2 PRODUCT
ComponentSource Awards 2009-10

BESTSELLING PRODUCT AWARDS
Awards 2009-10
ComponentSource
TOP 5 PRODUCT
ComponentSource Awards 2009-10

BESTSELLING PRODUCT AWARDS
Awards 2009-10
ComponentSource
TOP 25 PRODUCT
ComponentSource Awards 2009-10

BESTSELLING PRODUCT AWARDS
Awards 2009-10
ComponentSource
TOP 50 PRODUCT
ComponentSource Awards 2009-10

Microsoft GOLD CERTIFIED Partner

Microsoft System Center Alliance Partner

Optimized for Visual Studio

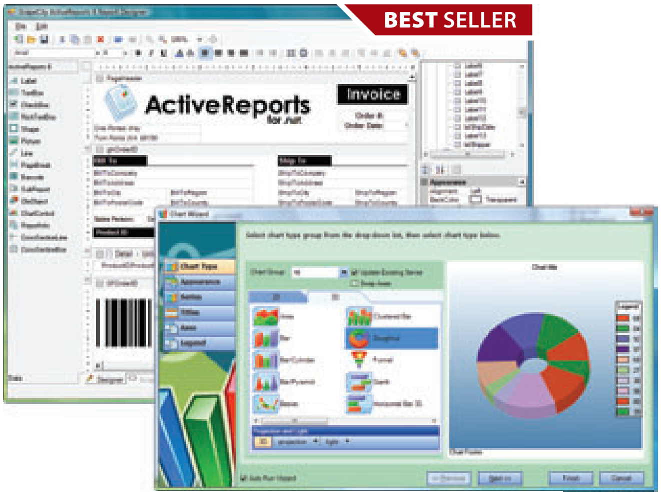
WINNER

GrapeCity

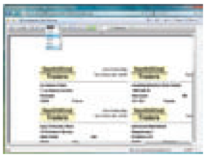
ActiveReports 6

The best selling report writer with no runtimes or royalties.

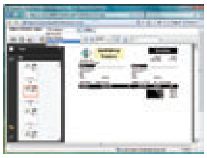
- Professional Edition features a royalty-free End-User Report Designer
- Includes filters for exporting to popular formats like PDF, Excel, RTF, HTML, Text and TIFF in both Windows- and Web-based applications



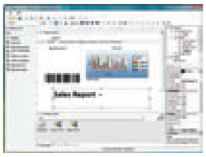
ActiveReports



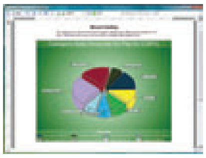
ViewerA nnotations
Annotations can be placed on the report to pass notes and special instructions on to users and developers.



End-User Report Designer Editing
Allow end users to edit their reports and save new reports or load old ones.



Viewing WebR eports
The WebViewer control provided in the Professional Edition allows you to write less code and view reports more quickly.



Built-In Charting Control
A chart control is included with ActiveReports that supports common 2D and 3D chart types.

Export to Any Format
ActiveReports includes filters for exporting to popular formats like Adobe PDF, Microsoft Excel, RTF, HTML, Text and TIFF.

Hyperlinks and Drilldown
You can use hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another.

Generate a wide variety of reports from your .NET applications.

ActiveReports 6 is a .NET reporting component for Windows and Web applications. Its key features include customization, fast performance and high quality. ActiveReports 6 features an easy-to-use Visual Studio .NET report designer and a powerful API. It includes multilanguage support that works right out of the box and also offers seamless run-time deployment, royalty free. It's easy to create reports in ActiveReports 6, the report designer integrates into Visual Studio .NET. Once the product has been installed on the developer's machine, adding a report to your project is as easy as adding a class or form.

ActiveReports 6 is easy to deploy - ActiveReports 6 reports are created within Visual Studio .NET and compiled directly into the executable. Therefore, assemblies can be distributed using XCopy deployment or they can be placed in the global assembly cache (GAC). The object model provides the Reporting Engine as a single, managed, strong named assembly. There are no additional setups on the server or the client's machine. Since ActiveReports 6 is fully managed, there are no dependencies on 3rd party applications.

ActiveReports 6 can export to many formats - it includes filters for exporting to popular formats like Adobe PDF, Microsoft Excel, RTF, HTML, Text and TIFF in both Windows- and Web-based applications.

ActiveReports 6 contains a built-in charting control which supports common 2D and 3D chart types and provides advanced charting features as well as native exporting to various image formats. Chart types include: vertical and horizontal bar, line, scatter plot, pie, funnel and pyramid, Gantt, Kagi, point and figure, renko, and more.

ActiveReports 6 supports the use of Windows Forms-based 3rd party controls with the report designer. Essentially any control that is able to be rendered on a Windows Form is capable of being rendered on an ActiveReports 6 report.

ActiveReports 6 includes a report viewer control and an End-User Report Designer - The report viewer control supports report zooming and previewing, multiple tabs for viewing hyperlinks, split- and multi-page views, a table of contents pane, thumbnails, text searches, annotations, and toolbar customization. ActiveReports 6 also features a royalty-free End-User Report Designer control that allows you to host the report designer in your own applications to provide end users with the ability to create and modify reports. The Zoom toolbar item allows users to magnify the design surface of the report for more precise placement and viewing of report controls. The Professional Edition of ActiveReports 6 also includes a Flash-based Report Viewer that you can use to provide a seamless, interactive report viewing experience and no-touch printing using the widely-adopted Adobe Flash Player. Choose from pre-defined themes (including Office, Windows XP, Vista and Windows Classic) and enhance the viewer experience by controlling the printing, paging, table of contents, thumbnail views and other options.

ActiveReports 6 comes with a Microsoft Access and Crystal report conversion wizard that makes importing Access or Crystal reports a breeze. This is a very handy utility for users who have invested a lot of time creating reports within Access or Crystal. Not only will the import help get you up and running with ActiveReports 6, but the banded report architecture is very familiar for developers that have used Microsoft Access' reports.

FROM GRAPECITY

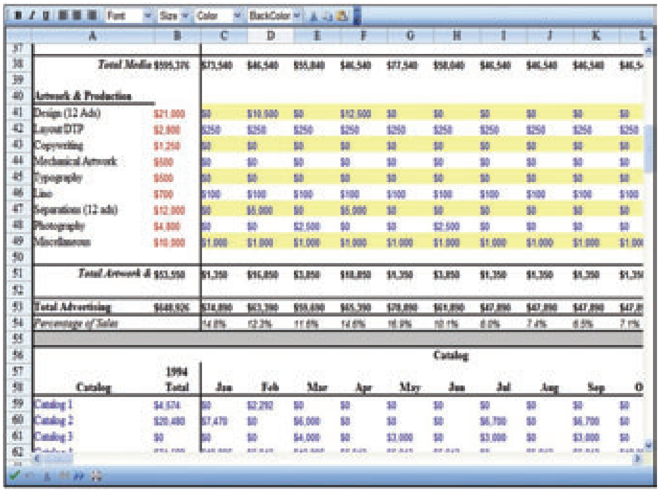
Product	Top 250	Review	Price	Product	Top 250	Review	Price
ActiveReports 6	7	-	$685.02	ActiveReports2f or ActiveX/COM V2.0	-	★★★★	$489.02
ActiveReportsf or. NET V3.0	248	★★★	$569.05	ActiveReports Reporting/BI Suite for .NET V1.0	-	-	$2,351.02
ActiveAnalysis 2	-	-	$979.02	DataD ynamics Reports V1.6	-	-	$1,077.02
ActiveBIS uitef or. NET V1.0	-	-	$1,665.02				

GrapeCity

Spread for ASP.NET V5.0

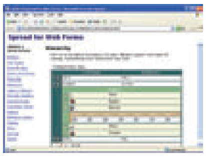
High-performance, fully-customizable ASP.NET spreadsheet component.

- Outlook style grouping allows users to sort data as required
- Conditional formatting enables styling based on cell contents
- Supports "Load on Demand" paging (virtual mode)

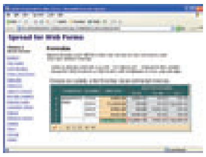


	A	B	C	D	E	F	G	H	I	J	K	L
37												
38	Total Media	$595,376	$73,540	$46,540	$55,040	$46,540	$77,540	$58,040	$46,540	$46,540	$46,540	$46,5
39												
40	Artwork & Production											
41	Design (12 Ads)	$21,000	$0	$10,500	$0	$12,500	$0	$0	$0	$0	$0	$0
42	Layout DTP	$2,800	$250	$250	$250	$250	$250	$250	$250	$250	$250	$250
43	Copywriting	$1,250	$0	$0	$0	$0	$0	$0	$0	$0	$0	$0
44	Mechanical Artwork	$500	$0	$0	$0	$0	$0	$0	$0	$0	$0	$0
45	Typography	$500	$0	$0	$0	$0	$0	$0	$0	$0	$0	$0
46	Lino	$700	$100	$100	$100	$100	$100	$100	$100	$100	$100	$100
47	Separations (12 ads)	$12,000	$0	$6,000	$0	$6,000	$0	$0	$0	$0	$0	$0
48	Photography	$4,800	$0	$0	$2,500	$0	$0	$2,500	$0	$0	$0	$0
49	Miscellaneous	$10,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,00
50												
51	Total Artwork &	$53,550	$1,350	$16,850	$3,850	$18,850	$1,350	$3,850	$1,350	$1,350	$1,350	$1,35
52												
53	Total Advertising	$648,926	$74,890	$63,390	$58,890	$65,390	$78,890	$61,890	$47,890	$47,890	$47,890	$47,8
54	Percentage of Sales		14.8%	12.2%	11.6%	14.6%	16.9%	10.1%	8.0%	7.4%	6.5%	7.1%
55												
56							Catalog					
57		1994										
58	Catalog	Total	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	O
59	Catalog 1	$4,574	$0	$2,292	$0	$0	$0	$0	$0	$0	$0	$0
60	Catalog 2	$20,480	$7,470	$0	$6,000	$0	$0	$0	$6,700	$0	$6,700	$0
61	Catalog 3	$0	$0	$0	$4,000	$0	$3,000	$0	$3,000	$0	$3,000	$0
62	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

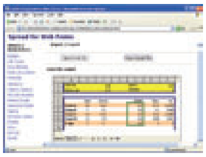
Spread for ASP.NET



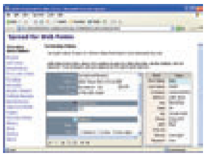
Hierarchy
Easily display and edit your hierarchical data. You have full customization at the child level if you want to hide columns etc.



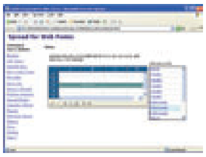
Formulas
Add calculations quickly to your Web applications by using any of over 200 predefined algorithms or add your own custom functions.



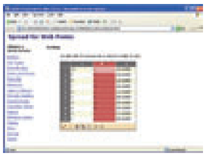
Import/Export
Import/export Microsoft Excel 97 and higher files and text files. You can also export to Spread XML.



Retrievingd ata
Retrieve data from bound and unbound sources easily.



Skins
Use any of the 13 predefined skins or save and load your own design. Customized skins can easily be shared within your dev team.



Sorting
Automatically sort any column in ascending or descending order by double clicking on the header.

Easily present, edit and update your ASP.NET data.

FarPoint Spread for ASP.NET is a spreadsheet component that offers intuitive customer usability along with a wealth of grid functionality. With Spread for ASP.NET you have exceptional client-side validation to alert your user of invalid data immediately and simple client-side keyboard navigation. Written from the ground up in C#, Spread for ASP.NET will provide your user with an easy and reliable, feature-rich, user-friendly online experience.

FarPoint Spread for ASP.NET offers excellent design time support for rapid development using the innovative Spread Designer and supports in-cell editing and validation. It supports bound or unbound modes (no dataset needed), AJAX, native Microsoft Excel import/export, in-cell editing, multiple edits on multiple rows without server round trips, client-side column/row resizing, multiple sheets, searching, filtering, multiple cell types, validations, cell spans, multiple headers, sorting, complete customization at the cell level, over 300 built-in calculation functions, hierarchical display and much more.

The main features of FarPoint Spread for ASP.NET include:

AJAX Support - AJAX allows the component to refresh without refreshing the entire page. This affects several features including expanding and collapsing child sheets in a hierarchical display, column sorting, inserting rows, searching, filtering, paging and formulas.

Grouping - You can set up the display to allow Outlook-style grouping of rows. For large amounts of data, this is helpful to display the data in the order the user needs. The user selects columns by which to sort and the component then organizes and displays the data in a hierarchy with rows organized accordingly.

Searching - You can search for data in any of the cells in the workbook by specifying the sheet and the string of data for which to search. You also have several searching options you can choose from.

Row Filtering - With row filtering, you can allow the user to filter the data in columns on a sheet and display only the rows of data which meet criteria from a drop-down list or change the appearance of rows based on that filtering.

Frozen Rows and Columns - You can freeze columns and rows and keep them displayed regardless of where the user navigates in the sheet. Frozen rows and frozen columns do not scroll when the user uses the scroll bar or navigation keys in the component.

Load on Demand - You can allow the Web page to load on demand - as the user scrolls further down the spreadsheet, the FarPoint Spread component on the client loads another page of rows from the server as needed. This is similar to what was called "virtual mode" in the COM version of the product.

Publish Excel Workbooks Online - You can easily publish your existing or new Excel workbooks online by using Spread's Excel import and export feature.

FROM GRAPECITY

Product	Top 250	Review	Price
Spreadf or WindowsF orms 5	13	-	$959.04
Spreadf orASP.NET V5.0	65	-	$979.02
Spread for .NET Bundle	197	-	$1,567.02
FarPointS pread V8.0	244	★★★★	$671.04
Input Pro for Win Forms	-	-	$244.02
ButtonO bjx V2.0	-	-	$195.02
CalendarO bjx V3.0	-	-	$224.42

Product	Top 250	Review	Price
Enterprise Database Pack	-	-	$881.02
EnterpriseD eveloper Toolbox	-	-	$979.02
InputP ro V3.0	-	-	$146.02
Input Pro for Silverlight	-	-	$293.02
ListP ro V3.0	-	★★★★★	$240.10
TabP ro V3.1	-	-	$126.42

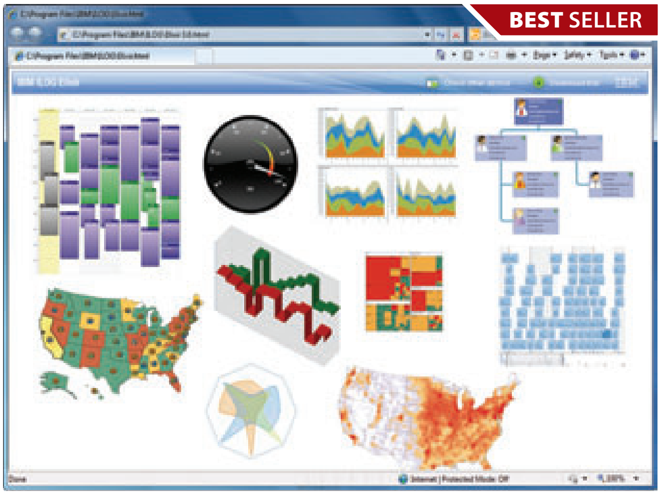
Performance for the Month
December, 2009
COMMENTARY
SALES
NEW CUSTOMERS
CUSTOMERS
Catalog Budget

IBM ILOG

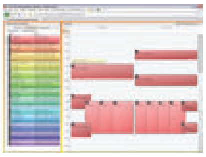
IBM ILOG Elixir Standard V3.0

A collection of innovative display components for Flex user interface developers.

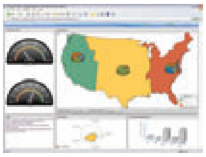
- Market leading data visualization component set for Adobe Flex 4 and Adobe AIR rich Internet application (RIA) developers
- Provides: 3D charts, gauges, maps, calendars, OLAP & pivot charts and more



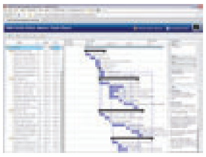
IBM ILOG Elixir Standard



Calendar
Easily display multiple planning at once together with advance recurrence management.

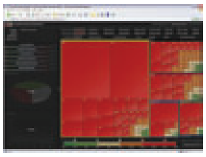


VectorM aps
Map areas are colored to represent specific data and overlaid with any Flex object, such as charts and labels.



Charts
IBM ILOG Elixir comes with a full range of charts.

Gauges and Dials
Gauges and dials are fully interactive and designed to be connected to real-time data sources.



Treemaps
Visually detect trends and outliers in large data sets.

OLAP and Pivot Charts
Replace OLAP grids with highly dynamic charts for improved insight and analysis in your custom data.

Data-display components that help turn raw data into clear, actionable information through a highly graphical and interactive user experience.

IBM® ILOG Elixir® Standard provides Flex user interface developers with a collection of innovative display components. It consists of 10 graphical components that present data to users so that they can understand the information more clearly, react faster and make better decisions.

IBM® ILOG Elixir® Standard provides 10 ready-to-use user interface controls: 3D charts, gauges, maps, heat maps, calendars, OLAP and pivot charts, organization charts, treemaps, radar charts, and timelines. IBM® ILOG Elixir® Standard's controls can be used individually or in combination, alongside other Flex components and they integrate with the Adobe Flash Builder 4 IDE.

The professionally documented software development kit (SDK) enables developers to fine tune the look or behaviors of the controls.

Adobe Flex and Adobe AIR form a powerful platform for creating graphically rich and highly interactive applications for the Internet (and desktop). It provides a wide variety of basic user interface components such as buttons, menus and charts. Many applications require user interfaces that are highly graphical, going beyond what is offered natively in the Adobe platform. IBM® ILOG Elixir® Standard enhances the Flex and AIR platforms by adding a set of advanced user interface controls for more intuitive, interactive displays.

Included controls:

3D Charts - IBM® ILOG Elixir® Standard comes with a full range of 3D charts that mirror the Adobe 2D charts. Upgrading to the third dimension and creating more appealing dashboards and custom applications has never been so easy.

Gauges and Dials - Gauges and dials are fully interactive and designed to be connected to real-time data sources. Developers can use them "as is", customize them or easily create new ones with the powerful and open APIs.

Vector Maps - Easily add intuitive map displays to your developments. Reuse one of the provided maps or create custom ones from standard maps you already have. You can easily color map areas to represent your data and overlay these areas with any Flex object, such as charts and labels.

Heatmaps - Heatmaps help you display how your data is spread over geographies or a Web page for instance. Manipulate either x/y or lat/lon and see your data by density or by value.

Calendar - Easily create custom and shared planning applications with the IBM® ILOG Elixir® Standard calendar. Display multiple planning with advanced recurrence management.

Timeline - Create easy to read charts displaying a sequence of events over a span of time. leverage the built-in overview that allows intuitive navigation.

Organization Charts - Advanced employee organization charts are easy to create with IBM® ILOG Elixir® Standard. Quickly navigate through peers and management relationships with stunning animations.

OLAP and Pivot Charts - Replace OLAP grids with highly dynamic charts for improved insight and analysis in your custom data.

Radar Charts - IBM® ILOG Elixir® Standard radar charts are compact dashboard displays that are used to compare different key performance indicators (KPIs) between two or more entities.

Treemaps - Treemaps provide an interactive experience that includes dynamic drill down, user-defined clustering and segment coloring and sizing.

FROM IBM ILOG

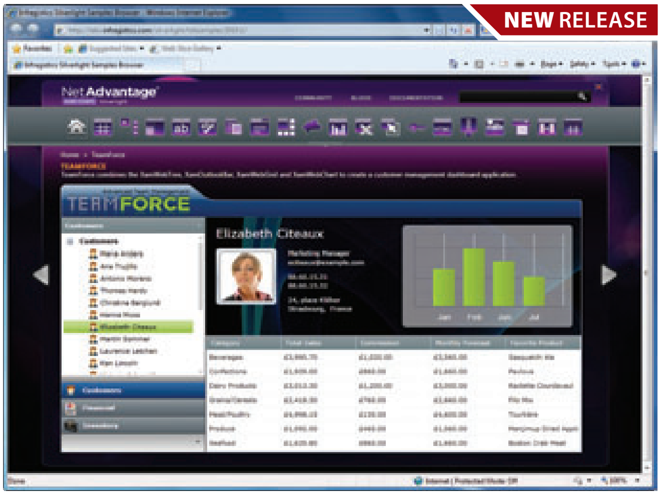
Product	Top 250	Review	Price
IBM ILOG Elixir Standard V3.0	11	-	$999.00

Infragistics

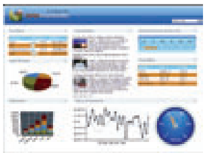
Infragistics NetAdvantage for .NET 2011 Volume 1

Create commercial class UIs for Winforms, ASP.NET, WPF & Silverlight applications.

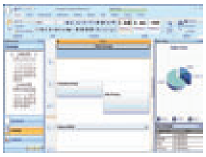
- Comprehensive User Interface control and component toolset for .NET
- Office 2010 File Support for ExcelML files allows you to export the contents of WebDataGrid and WebHierarchicalDataGrid.



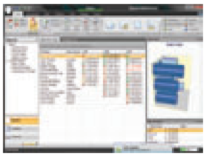
Infragistics NetAdvantage for .NET



NetAdvantage for ASP.NET
Gauges and Chart controls let you create an executive dashboard.

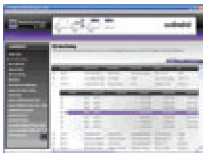


NetAdvantagef or Windows Forms
Create Outlook style applications with integrated NetAdvantage for Windows Forms controls.

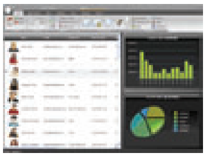


Create Applications with Office 2010 Styling
Also includes grids with conditional formatting, trees, 3D charts, menus, etc.

PDFEx port
You can export reports to PDF documents displaying gauges to visually summarize data.



DataGrid
Create a WPF grid using xamDataGrid, xamCarouselListBox and xamCarouselPanel.



WPFC harts
Display values using columns, pie wedges, cylinders and more using the advanced graphics rendering capabilities in the WPF platform.

Includes essential User Interface controls for Windows Forms, WPF, Silverlight and ASP.NET, plus application styling.

Infragistics NetAdvantage for .NET is a presentation layer toolset for quickly and easily building and styling polished application UIs and delivering a great user experience, for WPF, Silverlight, Windows Forms, ASP.NET and Tablet PC. You can deliver user experiences consistent with Microsoft Office and Windows, produce high performance Web applications and create hi-fidelity, cross-platform charts and gauges that instantly provide your audience with a clear understanding of key performance indicators. Designed with the user experience in mind, NetAdvantage for .NET user interface controls give you the ability to focus on business logic while it takes care of the presentation. Use the modern look & feel of Windows 7 styles and navigation or emulate the Office 2003, 2007 and 2010 user experiences with predefined color tables and ribbon controls.

Infragistics NetAdvantage for .NET has what you need to add no-touch AJAX to your Web site with a professionally polished UI. It includes many Windows Forms and ASP.NET components: grids, charts, combos, desktop alerts, buttons, calculator, dropdown, progress bar, scroll bar, tooltips, editors, gauge, listbar, listview, printing, scheduling, spell checker, tabs, toolbars and trees.

Infragistics NetAdvantage for .NET also includes a Document Export Engine for PDF and XPS formats and provides Web crawler optimization, AJAX support and SharePoint integration.

Infragistics NetAdvantage for .NET includes Silverlight Controls for Line of Business (LOB) Applications which enable you to include islands of richness into your existing Web-based applications, giving end-users an experience that goes beyond what can be accomplished with mere HTML. These controls have been built from the ground-up using the Silverlight platform and take full advantage of its capabilities to help you produce compelling Web applications. NetAdvantage for .NET includes next generation WPF controls which were designed from the start with an uncompromising attitude toward leveraging its cinematic graphics, rich styling, animation capabilities, and integrated UI virtualization. Embracing templates and exposing numerous styling points gives you complete command over the controls' appearance. A logical, friendly object model makes it easy to take advantage of the WPF platform's potential, whether you choose to do so through declarative UI (XAML) or code.

Multiple volume releases throughout the year continuously add valuable new features ensuring the tools available to you remain leading edge.

FROM INFRAGISTICS

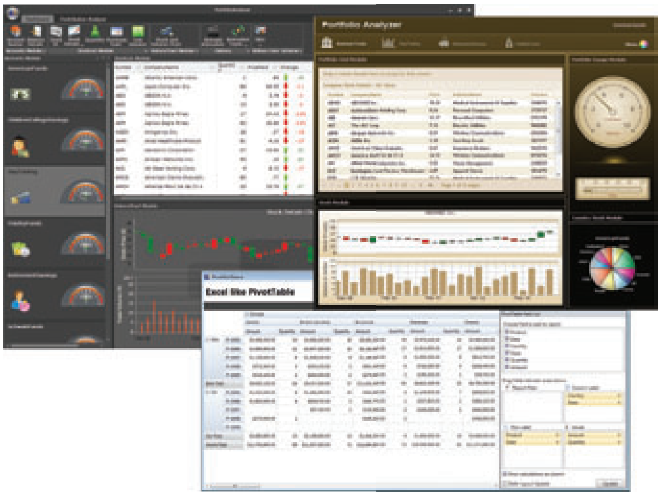
Product	Top 250	Review	Price
Infragistics NetAdvantage for .NET 2011 Volume 1	2	★★★★	$1,139.60
Infragistics NetAdvantage for Win Client 2011 Volume 1	5	-	$399.71
Designer Widgets V2.0	19	★★★	$295.00
Infragistics NetAdvantage Select 2011 Volume 1	23	★★★★	$399.71
NetAdvantageU ltimate 2011 Volume1	33	-	$1,800.25
Infragistics NetAdvantage for Web Client 2011 Volume 1	34	-	$399.71
Infragistics NetAdvantage for Silverlight Data Visualization 2011 Volume 1	46	-	$975.10
Data Widgets V3.1	78	★★★	$395.00
Infragistics TestAdvantage for Windows Forms (for HP QuickTest Professional software)	94	-	$2,935.10
Calendar Widgets V1.1	188	★★★	$295.00
UltraSuite V3.02	211	-	$495.00
NetAdvantage for Silverlight Line of Business 2011 Volume 1	237	-	$975.10
ActiveThreed Plus V3.0	239	-	$295.00
NetAdvantageI CONS Software&C omputingP ack V1.0	-	-	$195.02
CodeAssist V1.1	-	★★★★	$295.00
Infragistics NetAdvantage for WPF Line of Business 2011 Volume 1	-	-	$945.25
Infragistics NetAdvantage for WPF Data Visualization 2011 Volume 1	-	-	$975.10
Infragistics TestAdvantage for Windows Forms (for IBM Rational Functional Tester)	-	-	$2,935.10
QuinceP ro V1	-	-	$681.10
VBAssist V5.0	-	-	$179.00

Syncfusion

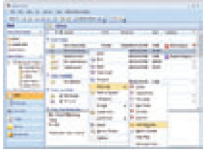
Syncfusion Essential Studio Enterprise Edition 2011

Accelerate Silverlight, WinForms, WPF, ASP.NET and ASP.NET MVC development.

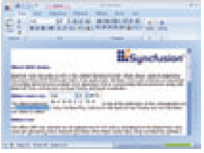
- Includes components for grids, docking, charting, diagramming, PDF & Office reporting and scheduling
- Easily build WPF and ASP.NET Business Intelligence applications



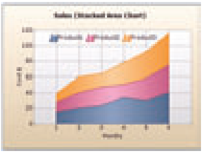
Syncfusion Essential Studio Enterprise Edition



OutlookU I
Create ASP.NET and Windows Forms applications that mimic the Outlook user interface.

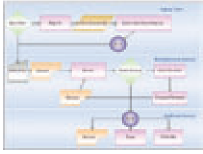


Ribbon
Silverlight Ribbon is a user interface library that allows users to create UIs similar to Office 2007.

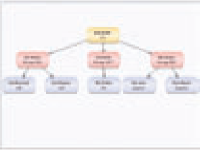


NumerousC hart Types
Includes support for Enterprise quality chart creation capabilities with all the chart types you can think of.

Support for Financial Charts
Create advanced financial charts like the stock-volume chart you see above.



StaticD iagrams
Use the Diagramming package to create Visio like diagrams and render them in your applications.



DynamicD iagrams
You can also create diagrams dynamically based on relationships in the database.

Syncfusion Essential Studio Enterprise is a dynamic and feature rich suite that has everything developers need to create Enterprise-class applications.

Syncfusion Essential Studio Enterprise includes all Syncfusion products on all technologies (WPF, Windows Forms, ASP.NET, ASP.NET MVC and Silverlight) Syncfusion Essential Studio Enterprise incorporates a unique debugging support system that allows switching between Debug and Release versions of the library with a single click from inside the Visual Studio.NET IDE. Dynamic and feature-rich, Syncfusion Essential Studio has everything developers need to create enterprise-class applications.

Essential Grid (ASP.NET, ASP.NET MVC, Silverlight, Windows Forms, WPF) - Excel-like, cell-oriented, light-weight, virtual, data-bound grid. Includes extremely detailed customization down to cell level. Multilevel undo/redo, shared scrollbar support, data/view separation, floating cells, multiple cell and more.

Essential Tools (ASP.NET, ASP.NET MVC, Silverlight, Windows Forms, WPF) - A rich set of UI controls to let you create rich interactive applications with the latest look and feel.

Essential Chart (ASP.NET, ASP.NET MVC, Silverlight, Windows Forms, WPF) - A business-oriented charting component which offers an innovative data object model that makes it easy to populate a chart with any kind of data source.

Essential Diagram (ASP.NET, Silverlight, Windows Forms, WPF) - An extensible, high-performance .NET diagramming component that can be used for developing Microsoft Visio-like interactive graphics and diagramming applications.

Essential BI Grid (ASP.NET, Silverlight, WPF) - Use BI Grid to summarize, analyze, explore, and present summaries of critical data in your enterprise and make informed decisions. Users can easily drill down into all available levels of data and it can even be configured to drill down into relational data.

Essential BI Gauge (Silverlight, WPF) - provides users with the ability to display Key Performance Indicators (KPI) from an OLAP cube. KPIs can be displayed with goals and status visualizations to help create executive dashboards.

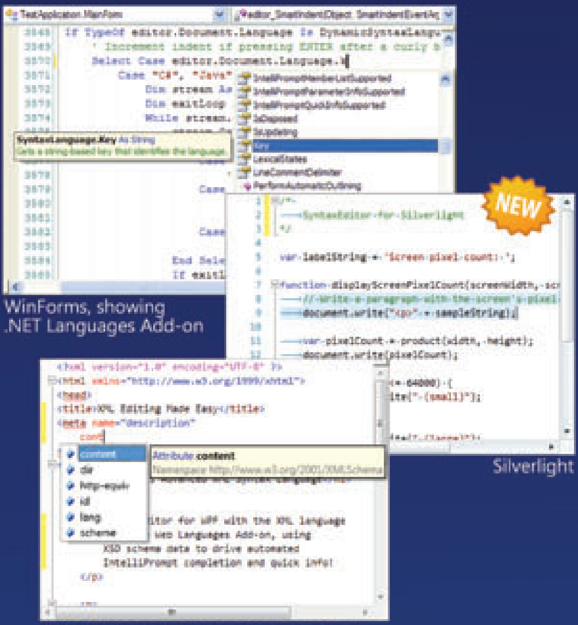
Essential BI PivotGrid (WPF) - A cell-oriented, extensible grid control that simulates the pivot table feature of Excel. It pivots data to organize it in a cross-tabulated form. The major advantage with the pivot grid is that you can extract the desired information from a large list within seconds. Along with presenting the data, the pivot grid also enables you to summarize and group data. It finds its main application in the financial domain, where it is used to organize and analyze business data.

FROM SYNCFUSION

Product	Top 250	Review	Price
Syncfusion Essential Studio Enterprise Edition 2011	61	★★★★	$1,955.10
Syncfusion Essential Studio User Interface Edition 2011	101	★★★★★	$1,661.10
SyncfusionE ssentialXls IO 2011	153	★★★★	$877.10
SyncfusionE ssentialG rid 2011	225	-	$877.10
SyncfusionE ssentialC alculate 2011	-	-	$877.10
SyncfusionE ssentialC hart 2011	-	-	$877.10
SyncfusionE ssentialD iagram 2011	-	★★★★★	$877.10
SyncfusionE ssentialD ocIO 2011	-	★★★★★	$877.10
SyncfusionE ssentialE dit 2011	-	-	$877.10
SyncfusionE ssential Gauge 2011	-	-	$877.10
SyncfusionE ssential Grid 2011	-	★★★★	$877.10
SyncfusionE ssential Grouping 2011	-	-	$877.10
SyncfusionE ssential HTMLUI 2011	-	-	$877.10
SyncfusionE ssential PDF 2011	-	★★★★★	$877.10
Syncfusion Essential Studio Business Intelligence Edition 2011	-	★★★★★	$1,661.10
Syncfusion Essential Studio Reporting Edition 2011	-	★★★★★	$1,661.10
SyncfusionE ssential Tools 2011	-	★★★★	$877.10

Actipro SyntaxEditor for WPF, Silverlight & WinForms

The ultimate syntax-highlighting code editor control



```
If TypeOf editor.Document.Language Is DynamicSyntaxLangu
    ' Increment indent if pressing ENTER after a curly b
    Select Case editor.Document.Language.K
        Case "C#", "Java"
            Dim stream As
            Dim exitLoop
            While stream.
```

TestApplication.MainForm
editor_SmartIndentObject SmartIndentEventArg
IntelliPromptMemberListSupported
IntelliPromptParameterInfoSupported
IntelliPromptQuickInfoSupported
IsDisposed
IsUpdating
Key
LexicalStates
LineCommentDelimiter
PerformAutomaticOutlining

SyntaxLanguage.Key As String
Gets a string-based key that identifies the language

WinForms, showing .NET Languages Add-on

```
/*
    SyntaxEditor for Silverlight
*/

var labelString = 'Screen pixel count: ';

function displayScreenPixelCount(screenWidth, scr
    // Write a paragraph with the screen's pixel
    document.write("<p>" + sampleString);

    var pixelCount = product(width, height);
    document.write(pixelCount);
```

Silverlight

```
<?xml version="1.0" encoding="UTF-8" ?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>XML Editing Made Easy</title>
<meta name="description"
    cont
```

Attribute content
Namespace http://www.w3.org/2001/XMLSchema

```
itor for WPF with the XML language
Web Languages Add-on, using
XSD schema data to drive automated
IntelliPrompt completion and quick info!
</p>
```

WPF, showing Web Languages Add-on

- Mimics the Visual Studio code editor
- Create custom syntax language definitions
- Lexing, parsing, AST generation, squiggle lines
- Code outlining and custom margins
- IntelliPrompt completion lists, quick info & more
- Over 100 stock editing commands

Optional Language Add-ons

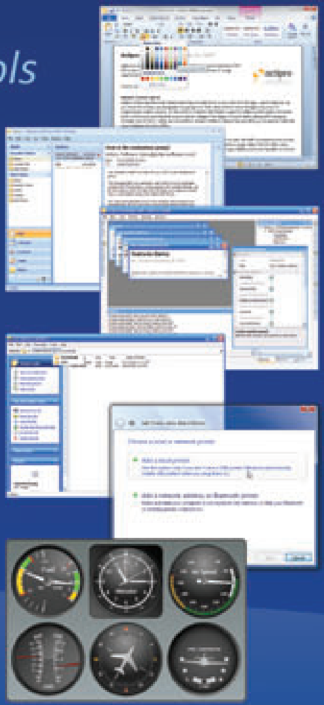
- Advanced C#, VB and XML languages
- Automated IntelliPrompt & validation
- Document outlining & auto-complete

Other Free Add-ons

- Innovative grammar-based parser framework
- Easy integration with ANTLR, Irony (WPF only)

Actipro WPF Studio

Jumpstart your WPF application's UI with over 55 controls



Bundled suite includes all Actipro WPF controls at one low price:

- **SyntaxEditor -** Syntax-highlighting code editor control like in VS 2010
- **Ribbon -** Office 2010 ribbon UI with Backstage application menu
- **Docking & MDI -** Docking windows, multiple document interface
- **Navigation** - Navigation/explorer bars, breadcrumbs, zoom UI
- **PropertyGrid** - Blend and VS-like property grid
- **DataGrid** - Themes, behaviors, editors integration w/MS DataGrid
- **Editors -** Part-based & masked editors w/advanced date/time picker
- **Wizard -** Aero wizard and wizard 97 dialogs
- **Gauge -** Circular, linear, and digital gauges
- **Bar Code -** Vector bar codes with many symbologies
- **Views -** Book control and animated panels with fluid transitions
- 1 year of support & free upgrades

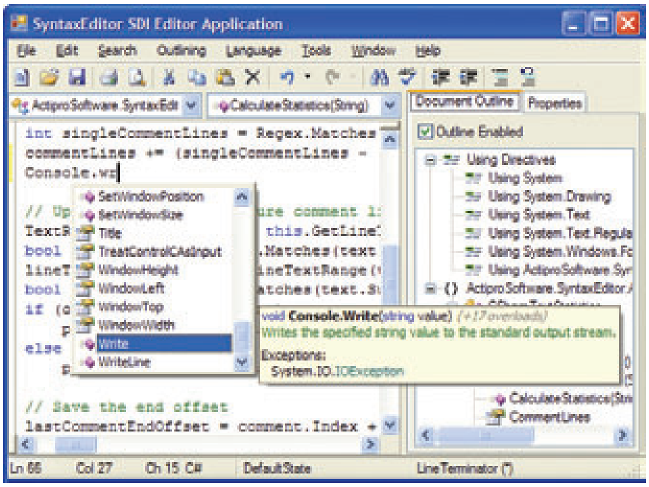
Discover the possibilities... free trials at:
componentsource.com/actipro

actipro

Actipro Software

Actipro SyntaxEditor V4.0

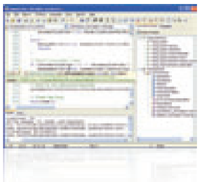
Powerful code editing control with VS.NET style productivity enhancements.



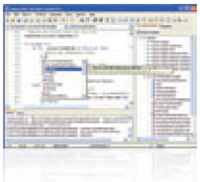
```
int singleCommentLines = Regex.Matches
commentLines += (singleCommentLines -
Console.wr
// Up                ure comment l:
TextR                 this.GetLine(
bool                 .Matches(text
lineT                ineTextRange(
bool                 atches(text.S
if (c
    P
else
    P
// Save the end offset
lastCommentEndOffset = comment.Index +
```

void Console.Write(string value) (+17 overloads)
Writes the specified string value to the standard output stream.
Exceptions:
System.IO.IOException

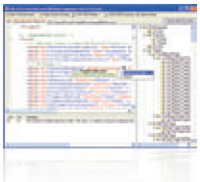
Actipro SyntaxEditor



Editing C# code
Edit C# code using automated IntelliPrompt for parameter information.



Editing VBc ode
Edit VB code using automated IntelliPrompt for member lists.



Editing an XML SyntaxEditor language definition file
SyntaxEditor can edit an XML SyntaxEditor language definition file based on an XSD.

- Develop and distribute custom language definitions with your applications
- Supports language merging e.g. HTML calling CSS, VBScript, etc.
- Perform semantic parsing operations in a separate worker thread

Actipro SyntaxEditor was designed to handle advanced language parsing and processing, with phases for performing lexical parsing, semantic parsing (in a worker thread), and automatic outlining. Custom language definitions can be developed and distributed with your applications. SyntaxEditor includes a complete parser generator that helps build semantic parsers and construct ASTs (abstract syntax trees). Major code editing features of the control include: code outlining, undo redo, word wrap, split views, current line and bracket highlights, virtual space, block selection, macro recording/playback, indicators, extensible rendering, IntelliPrompt code snippets, member lists, smart tags and more.

FROM ACTIPRO SOFTWARE

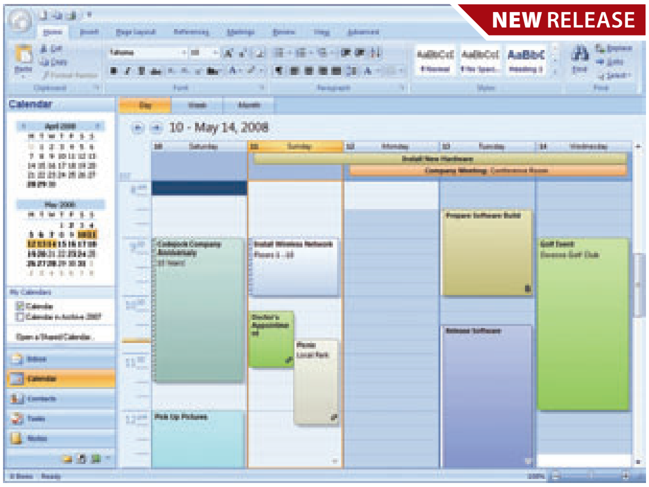
Product	Top 250	Review	Price
ActiproS yntaxEditor V4.0	133	★★★★★	$342.95
ActiproU IStudioC ontrolS uite V2.0	162	-	$293.95
Actipro Wizard V3.0	-	-	$88.15
Actipro Bar Code for WPF 2011	-	-	$97.02
Actipro Docking & MDI for WPF 2011	-	-	$195.02
ActiproE ditorsf or WPF 2011	-	-	$97.02
ActiproG augef or WPF 2011	-	-	$244.02
Actiprol cons V1.0	-	-	$293.02
ActiproNa vigationf or WPF 2011	-	-	$116.62
ActiproP ropertyGridf or WPF 2011	-	-	$146.02

Codejock Software

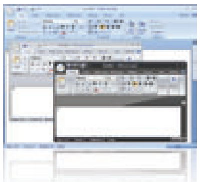
Codejock Xtreme Toolkit Pro

Create Office and Visual Studio .NET style GUI elements.

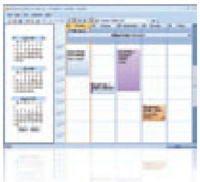
NEW RELEASE



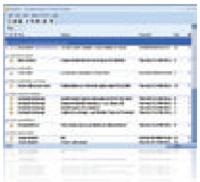
Codejock Xtreme Toolkit Pro



Ribbon
Create applications with the Microsoft Office 2007 style Ribbon Bar.



Calendar
Provides Windows developers with a sophisticated Outlook style calendar.



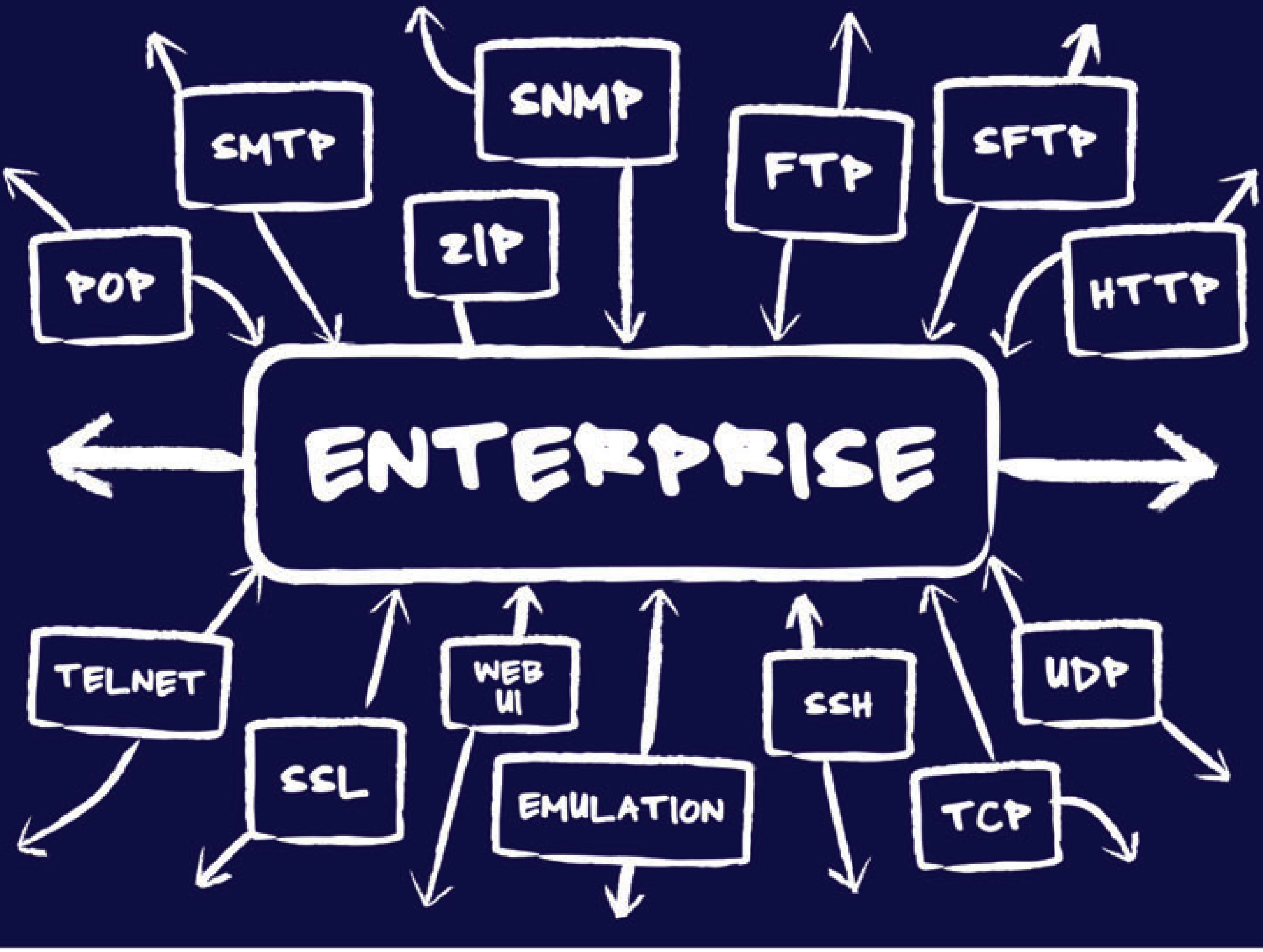
GridS orting
Easily create, group and sort data in a flat or hierarchical format.

- Produce Microsoft style ribbon, toolbars, menus and MDI tabs
- Easily theme your application with SkinFramework for .NET
- Includes an Enterprise class grid component with built in toolbar

Codejock Xtreme Toolkit Pro provides Windows developers with a sophisticated set of components to give applications a professional, modern appearance. Included are nine components: Xtreme Command Bars, Xtreme Docking Pane, Xtreme Property Grid, Xtreme Report Control, Xtreme Calendar, Xtreme Task Panel, Xtreme ShortcutBar, Xtreme SkinFramework and Xtreme Controls. With Codejock Xtreme Toolkit Pro you can create enterprise class applications that incorporate a full set of customizable user interface components that include Office style toolbars and menus. Several popular visual styles are included such as Office XP, 2003, 2007, 2010 and Visual Studio .NET.

FROM CODEJOCK SOFTWARE

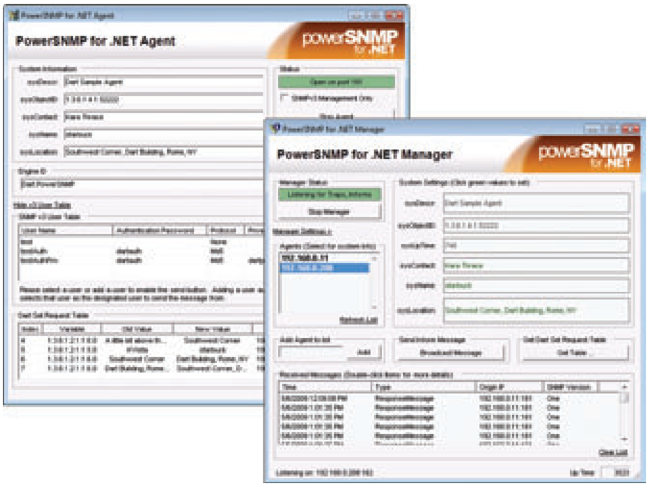
Product	Top 250	Review	Price
Codejock Xtreme Toolkit Pro	88	★★★★★	$502.55
Codejock Xtreme Suite Pro	165	★★★★★	$502.55
Codejock Xtreme Chart Pro	-	-	$224.42
Codejock Xtreme Calendar	-	-	$141.55
Codejock Xtreme Task Panel	-	-	$141.55
Codejock Xtreme Command Bars	-	-	$141.55
Codejock Xtreme Controls	-	-	$141.55
Codejock Xtreme Docking Pane	-	-	$141.55
Codejock Xtreme Property Grid	-	-	$141.55
Codejock Xtreme Report Control	-	★★★★★	$141.55


SNMP
SMTP
FTP
SFTP
ZIP
POP
HTTP
ENTERPRISE
TELNET
WEB UI
SSH
UDP
SSL
EMULATION
TCP

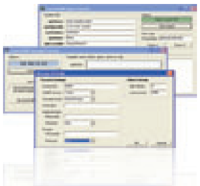
Dart Communications

PowerSNMP for .NET V4.2

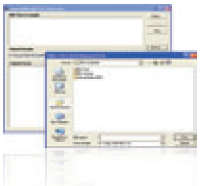
Manage, monitor and control network devices from your .NET applications.



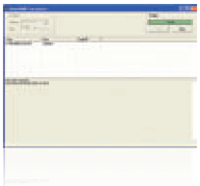
PowerSNMP for .NET



Agent
You can create an application with common SNMP agent functions such as responding to SNMP queries.



Generate MIB Classes
Generate class code for MIBs. This functionality is also built into the Manager and Agent components.



TrapC atching
Create NT services and contol them via the Windows services control panel.

- Messages can be constructed using no security, authentication only, or authentication and privacy with SNMPv3
- A Management Information Base compiler dynamically loads MIB files at runtime

PowerSNMP for .NET is a family of .NET classes and components used to manage network devices and remote management information. Full support for SNMPv1, SNMPv2, SNMPv3 and ASN.1 standards provide interoperability with Enterprise devices. The Manager and Agent components provide for message transport and a suite of supporting classes provide MIB (Management Information Base) information, encoding & decoding and encryption & decryption. MIB files can be parsed and imported into C# and Visual Basic projects to provide full intellisense support. A MIB compiler dynamically loads MIB files at runtime to generate new classes and objects on demand.

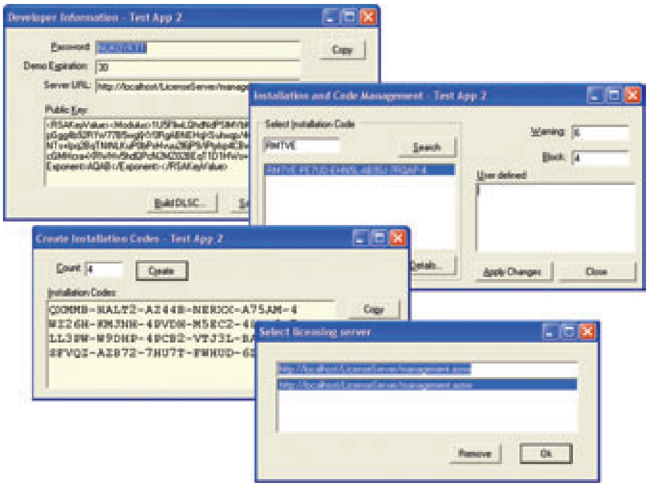
FROM DART COMMUNICATIONS

Product	Top 250	Review	Price
PowerTCPSSLf orA ctiveX V2.0	36	-	**$959.04**
PowerSNMPf or. NET V4.2	136	-	**$1,371.02**
PowerTCPF TPf or. NET V3.0	155	-	**$287.04**
PowerTCP for .NET and ActiveX Suite	200	-	**$2,111.04**
PowerTCPMai lf orA ctiveX V2.11	-	-	**$479.04**
PowerTCPF TPf orA ctiveX V2.1	-	★★★★	**$287.04**
PowerWEB File Upload for ASP.NET V1.3	-	★★★★★	**$146.02**
PowerSNMPf or. NETC F V4.2	-	-	**$1,371.02**
PowerSNMPf orA ctiveX V2.8	-	-	**$1,175.02**
PowerTCP .NET Suite Subscription	-	-	**$1,823.04**

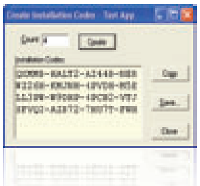
Desaware

Desaware Licensing System V2.0

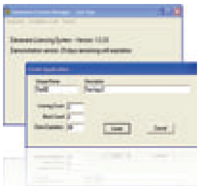
Add a cryptographic licensing system to your .NET applications.



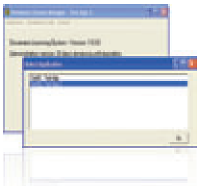
Desaware Licensing System



Create Installation Codes
You can create one or more installation codes. You can copy the codes to the clipboard or save them to disk.



Assemblies
Different types of assemblies across different products, can all be licensed with a single licensing certificate.



Applications elect
You can select and work on any application from a list of applications defined on the current server.

- Supports timed demo installations and server activation
- No hidden files, registry entries or background services needed
- License an unlimited number of applications at no extra cost

Desaware Licensing System is a cryptographic based licensing system for .NET. It is designed for component licensing per server/machine and is extremely easy to use. It can be configured for both moderate and high security scenarios. Using unique security features in the .NET Framework such as strong named assemblies and XML signatures, the Desaware Licensing System relies on end to end 128 bit cryptographic techniques, avoiding the use of hidden files, registry entries and other traditional licensing schemes.

Desaware Licensing System also includes an activation Web service, with a management application (with source).

FROM DESAWARE

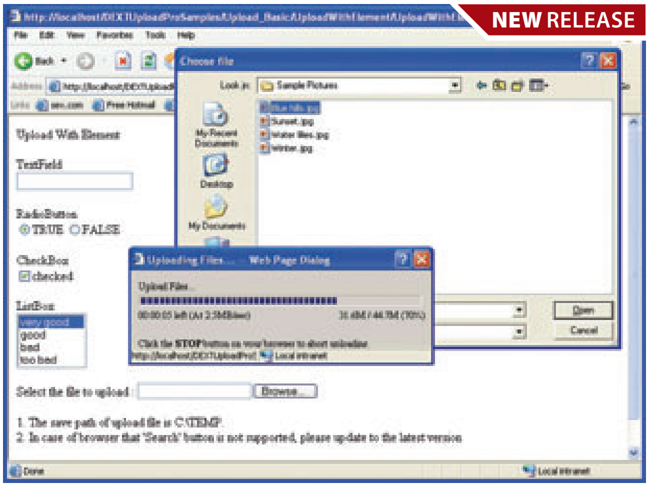
Product	Top 250	Review	Price
DesawareLi censingS ystem V2.0	-	★★★★★	**$1,465.10**
CAS/Tester V1.0	-	-	**$391.02**
DesawareA ctiveX Gallimaufry V2.0	-	-	**$143.04**
DesawareF ileP ropertyC omponent V1.0	-	-	**$77.42**
DesawareNT S ervice Toolkit V2.0	-	★★★★★	**$489.02**
Desaware NT Service Toolkit .NET Edition V2.0	-	-	**$479.04**
Desaware Universal .NET	-	-	**$636.02**
Desaware Universal COM	-	-	**$636.02**
EventL og Toolkit V1.0	-	-	**$146.02**
INIFileTool-5M V1.0	-	-	**$77.42**

US Elections - Make your predictions

DEVPIA

DEXTUpload .NET Pro V4.0

Add the ability to upload files to your Web applications.



NEW RELEASE

http://localhost/DEXTUploadProSamples/Upload_Basic/UploadWithElement/UploadWith...
File Edit View Favorites Tools Help
Back
Address http://localhost/DEXTUpload
Links msn.com Free Hotmail
Choose file
Look in: Sample Pictures
Blue hills.jpg
Sunset.jpg
Water lilies.jpg
Winter.jpg
My Recent Documents
Desktop
My Documents
Open
Cancel
Uploading Files... -- Web Page Dialog
Upload Files...
00:00:05 left (At 2.5MB/sec)
31.4M / 44.7M (70%)
Click the STOP button on your browser to abort uploading
http://localhost/DEXTUploadPro/ Local intranet
Done
Local intranet

Upload With Element

TextField

RadioButton
◉ TRUE ○ FALSE

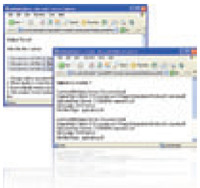
CheckBox
☑ checked

ListBox
very good
good
bad
too bad

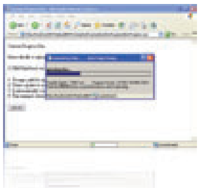
Select the file to upload : Browse...

1. The save path of upload file is C:\TEMP.
2. In case of browser that 'Search' button is not supported, please update to the latest version

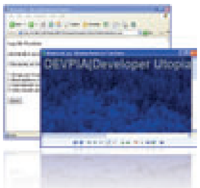
DEXTUpload .NET Pro



Single and Multiple Uploads
DEXTUpload .NET Pro allows you to upload single and multiple files.



UploadP progress
You can display a custom progress bar during the upload process.



Watermarking and Image Format
You can upload and watermark images and change image formats on the fly.

- Allows uploads of upto 2GB files
- Use of optimized parsing algorithm makes uploading speed even faster
- Able to upload files to a database via a BLOB (Binary Large Object)

DEXTUpload .NET Pro is a component that supports file uploads and downloads between a Web browser and an IIS Web server. It observes Internet standards (RFC 1867) and can be used in Microsoft ASP.NET.

DEXTUpload .NET Pro is HTTP-based and uploads files from a Web browser to a Web server quickly and efficiently. It overcomes the limitations of ASP.NET uploading (inconsistency for high capacity file uploads, server overloads and safety, etc.), is written in C# and provides you with improved upload speeds and stability. During upload and download you can invoke a progress window automatically without installing any other software. The pop-up progress window shows you uploading speed, file size, uploaded file size and time remaining. It can also be customized by using progress objects. DEXTUpload .NET Pro allows you to watermark uploaded images and convert uploaded images to thumbnails. It supports various file conversion formats (BMP, JPG, GIF etc.) and provides detailed information about image files.

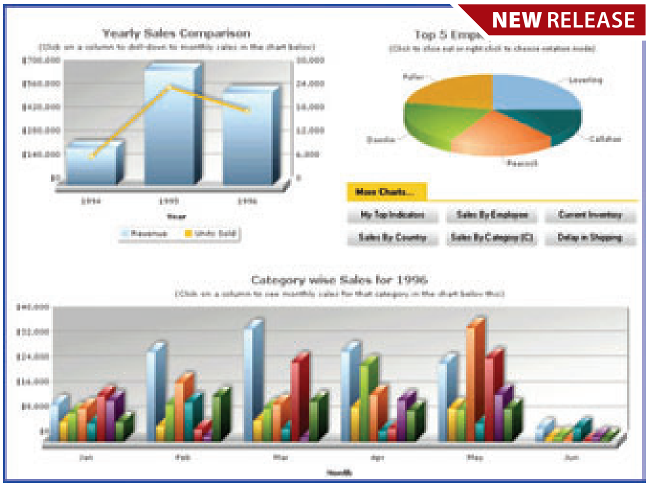
FROM DEVPIA

Product	Top 250	Review	Price
DEXTUpload. NETP ro V4.0	-	-	$331.24
DEXTUploadP ro V3.5	-	-	$293.02
DEXTUploadP roE xtension V1.5	-	-	$761.46
DEXTUploadJ V1.0	-	-	$369.46
DEXTUploadX V3.0	-	-	$369.46

InfoSoft Global

FusionCharts Bundle

Cross platform charting solution for ASP, ASP.NET, PHP, JSP & AJAX applications.

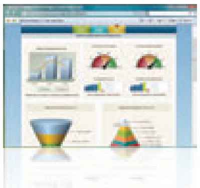


NEW RELEASE

FusionCharts Bundle

FusionMaps
Create animated, interactive and data-driven Flash Maps for your dynamic Web applications.

FusionCharts
Interactive and animated charts for ASP and ASP.NET applications.



FusionWidgets
Real-time Financial and Dashboard Charts for the Web.

- Generate interactive and animated flash charts & maps in minutes
- Supports over 75 chart types and 450 maps
- Build cross-browser and cross-platform solutions

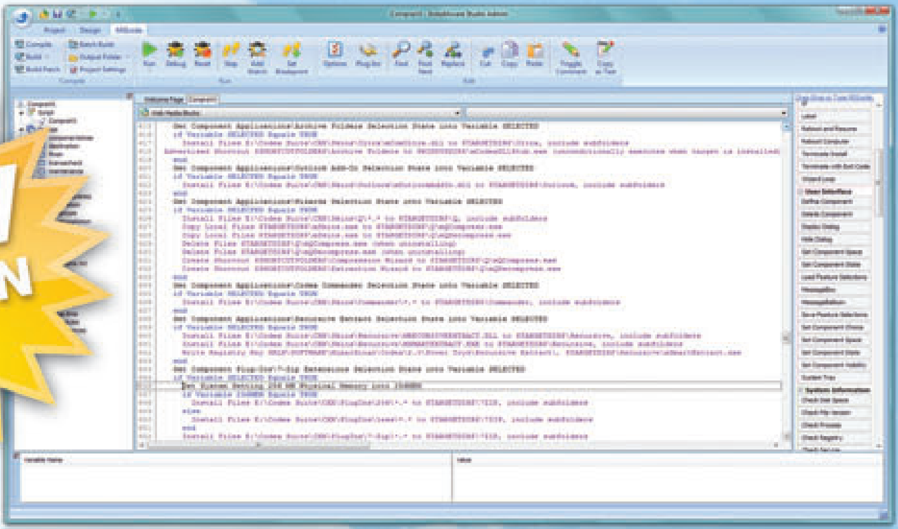
FusionCharts Bundled Pack includes: FusionCharts, PowerCharts, PowerMaps and FusionGadgets. The Pack provides developers with a collection of Flash components for creating highly visual Web applications based on the Adobe Flash Platform. Developers can build cross-browser and cross-platform solutions which blend content, application logic and communications delivering an effective end user experience. FusionCharts delivers interactive and eye catching Flash charts. FusionMaps can plot over 450 maps, including all continents, major countries and all US states. FusionWidgets is a collection of real-time gauges, self-updating charts and financial charts like Gantt charts, funnel/pyramid charts, bullet graphs, sparklines etc.

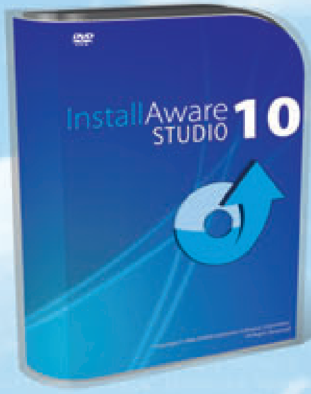
FusionCharts Bundled Pack can be used with any Web scripting language like .NET, ASP, JSP, PHP, ColdFusion, Ruby on Rails etc.

FROM INFOSOFT GLOBAL

Product	Top 250	Review	Price
FusionChartsB undle	26	-	$489.02
FusionCharts V3.2	70	-	$195.02
FusionChartsf orF lex V1.2	87	-	$293.02
FusionWidgets v3.2	230	-	$195.02
PowerCharts v3.2	234	-	$195.02
FusionChartsf or VB V1.0	-	-	$97.02
FusionMaps v3.2	-	-	$195.02

NEW
VERSION
10

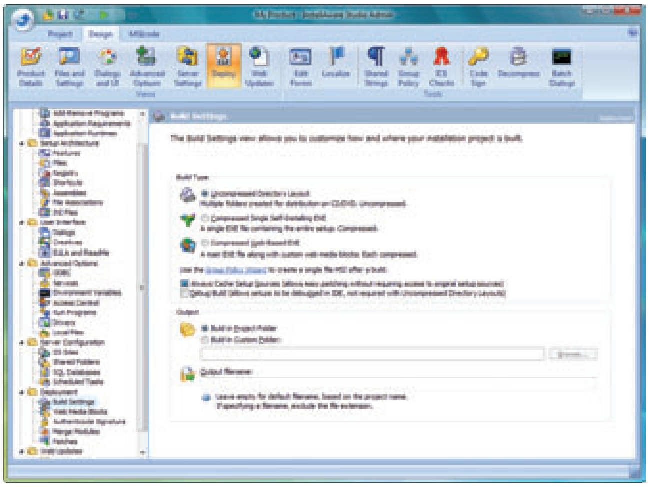



InstallAware 10
STUDIO

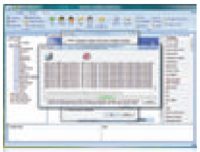
InstallAware

InstallAware Studio 10

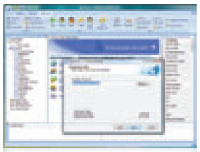
A setup authoring tool for ISVs and enterprises deploying products, patches and Web updates.



InstallAware Studio



One-Click Technology
Deploy technology modules and runtimes, such as Microsoft SQL Express and Crystal Reports.



DialogDes igner
InstallAware's dialog editor lets you customize the 15 pre-built setup themes, or design your own themes.

Import .MSI and .MSM Files
Directly import MSI and Merge Modules created by other packaging applications, and edit them within InstallAware.

- InstallAware setups are easily distributed through Group Policy Objects, and other software distribution systems
- Deliver setups that display the latest visual themes

InstallAware is a software installation solution for Windows Installer that enables MSIcode scripting for rapid setup development without the high cost and steep learning curve of other setup solutions. With InstallAware you can build sophisticated installations in record time without any programming or scripting skills. InstallAware provides you with the latest features and technology support, with the advantages of true rapid-Windows Installer development productivity.

InstallAware lets you set up 64 Bit and 32 Bit Systems with the Same Installer by detecting and supporting ia64 and x64 systems including Itanium, AMD64, and EM64T. InstallAware allows you to ship a single hybrid MSI for both your 32 bit and 64 bit customers.

InstallAware can check your target system and detect whether the operating system is running on a virtual machine. A simple checkbox lets you limit distributions to physical machines. It also makes it easy for your users to update their software via the Web, by automating the notification, download, and setup process.

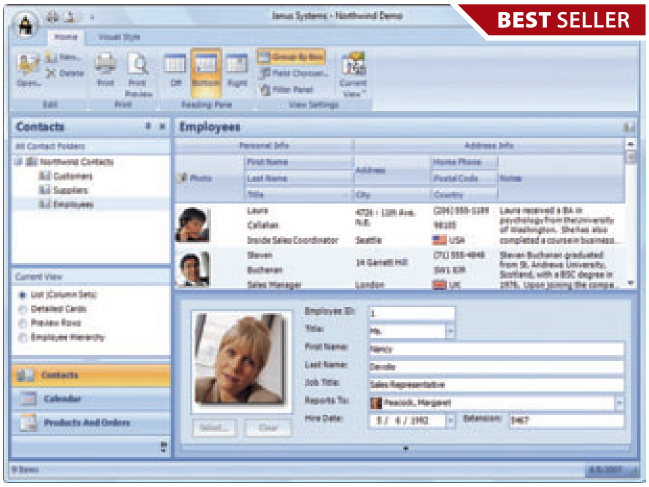
FROM INSTALLAWARE

Product	Top 250	Review	Price
InstallAwareS tudio 10	102	-	$1,959.02
InstallAwareD eveloper 10	-	-	$1,469.02
InstallAwareE xpress 10	-	-	$979.02
InstallAwareS tudioA dmin 10	-	-	$2,939.02

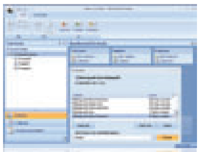
Janus Systems

Janus WinForms Controls Suite V3.5

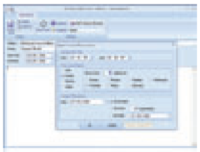
Add Outlook style interfaces to your .NET applications.



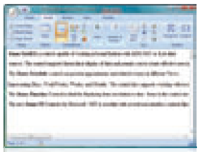
Janus WinForms Controls Suite



Office 2007
Janus WinForms Controls Suite offers an Office 2007 visual style for all controls included in the suite.



Janus Schedule Control for .NET
Present appointments and related owners in different views, representing Days, WorkWeeks, Weeks and Months.



RibbonC omponent
Janus WinForms Controls Suite includes a Ribbon control.

- Includes an Outlook style grid, calendar view, journal, Ribbon and shortcut bar
- Supports snap lines, base lines, ImageList keys and more
- Unbound mode in GridEX allows you to populate rows using AddItem method

Janus WinForms Controls Suite consists of 100% .NET managed code components that are designed to provide user interfaces for Microsoft .NET applications patterned after Microsoft Outlook.
Janus WinForms Controls Suite contains Janus GridEX for .NET (Outlook Grid), Janus Schedule for .NET (Outlook Calendar View), Janus Timeline Control for .NET (Outlook Journal), Janus ButtonBar Control for .NET and ExplorerBar Controls for .NET (Outlook Shortcut Bar), Janus CommandManager (Menus and Toolbars), Janus PanelManager (Outlook / VS.NET panes), Janus TabControl (tab control) and Janus Ribbon for .NET (Office 2007 style Ribbon bar). Janus GridEX for .NET is a data-aware, fully editable grid component that can be bound to any table in a dataset or to any other data source that supports IList, ITypedList or IBindingList interfaces. Janus GridEX for .NET does not sacrifice any group, sort or filter capabilities, when it is bound.

FROM JANUS SYSTEMS

Product	Top 250	Review	Price
Janus WinFormsC ontrolsS uite V3.5	17	★★★★	$757.44
JanusG ridEXf or .NET V3.5	91	★★★★	$469.44
Janus Web ASP.NET Server Controls Suite V3.0	126	-	$661.44
Janus Schedule and Timeline Controls for .NET V3.5	181	-	$431.04
JanusG ridEX 2000b	217	★★★★	$315.84
Janus UI Controls for .NET V3.5	-	★★★★★	$431.04

Janus WinForms Controls Suite

Add powerful Outlook style interfaces to your applications

Janus GridEX for .NET

Janus GridEX is a control capable of working in bound fashion with ADO.NET or IList data sources. The control supports hierarchical display of data and permits you to create effective user interfaces similar to Microsoft Outlook®, while easing development and maintenance time. The functionality provided by the control allows you to display, format, edit, sort, group, filter, manipulate, summarize, preview and print your data. Standalone MaskEdit, MultiColumn Combo, and UpDown controls are also included.

Janus Ribbon & Command Manager Controls

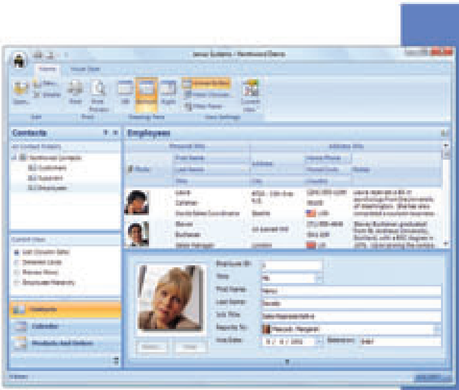
Janus Command Manager and new Ribbon controls are also included in the suite. These controls provide Windows developers with stylish menus and toolbars. The components permit advanced customization of the menus and toolbars of your applications. Use the built in VisualStyles in the controls to easily provide your applications with a look similar to Windows XP, Office 2000, Office XP, Office 2003 or 2007.

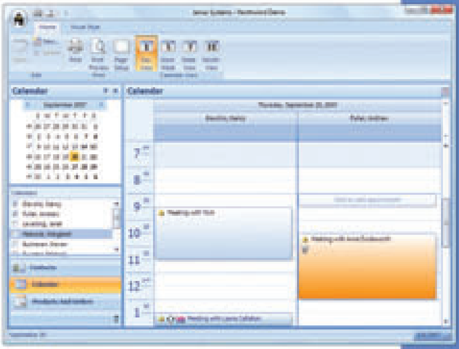
Janus Timeline Control for .NET

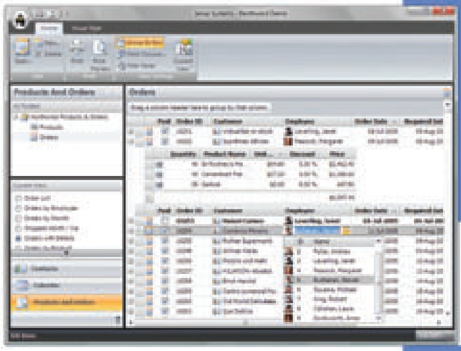
The Janus Timeline Control is ideal for displaying items in relation to time. Items in the control are arranged in chronological order from left to right on a time scale that can be divided from seconds to years. You can easily load, display, format, group, filter, edit and manipulate items with the functionality available in the control in both bound and unbound modes.

Janus Schedule for .NET

The Janus Schedule control can present appointments and related owners in different Views, representing Days, WorkWeeks, Weeks, and Months. The control also supports working with recurrent appointments. You can easily load, display, format, filter, edit, manipulate, preview, and print appointments with the functionality available in the control in both bound and unbound modes. A stand-alone Janus Calendar Combo Control, used to edit date and time values, is also included.







Windows Forms Controls

A suite of controls designed to provide powerful user interfaces for Microsoft .NET Smart Client applications patterned after Microsoft Outlook®. All of the controls in the suite are 100% C# managed controls designed to be used with Visual Studio® 2005.

Janus WinForms Controls Suite includes the following controls:

Janus GridEx for .NET
(Outlook Grid)

Janus Schedule for .NET
(Outlook Calendar View)

Janus Timeline for .NET
(Outlook Journal)

Janus ButtonBar for .NET
(Outlook Shortcut Bar)

Janus ExplorerBar for .NET
(Outlook Shortcut Bar)

Janus UI CommandManager
(Menus and Toolbars)

Janus PanelManager
(Outlook / Visual Studio .NET panes)

Janus TabControl
(Tab Control)

NEW!

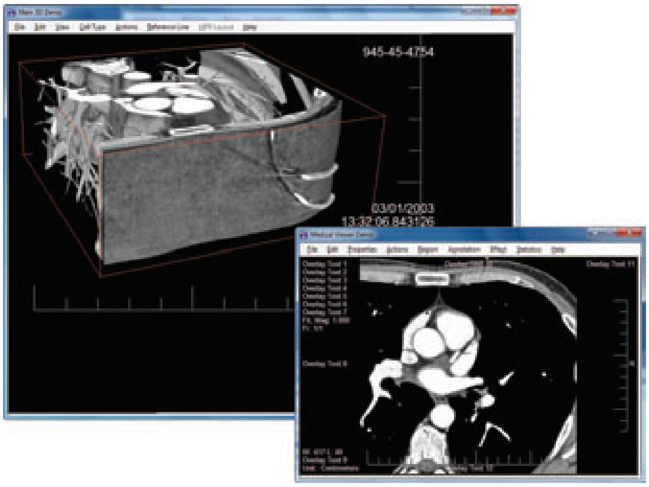
All controls now feature Office 2007 look & feel plus Ribbon Bar

Download a free trial or buy now at
www.componentsource.com

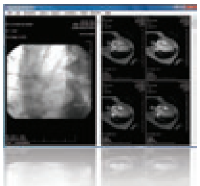
LEAD Technologies

LEADTOOLS Medical Imaging Suite SDK V17.0

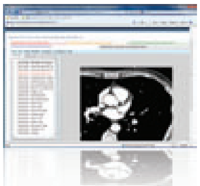
Medical imaging libraries for developers.



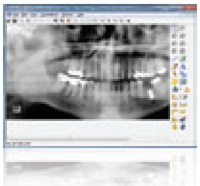
LEADTOOLS Document Medical Suite SDK



Imagehandli ng
Optimized internal core functions to dramatically increase the speed of image handling and processing.



3DR endering
Construct and view a 3D volume from a set of homogenous 2D medical images.



Windowsand Webde velopment
Includes native 32 and 64 bit PACS and medical imaging binaries.

- Develop powerful PACS and Medical imaging applications
- Development tools and components for Silverlight, Win Forms, Web Forms, WPF, C and C++ development

Optimized features to meet the specific needs of medical imaging application development. LEADTOOLS Medical Imaging Suite SDK features Medical Web Viewer Framework, Medical 3D, Medical Multimedia, high and low level PACS SCP and SCU functions and controls, DICOM PACS communication security, comprehensive full DICOM dataset support, 8-16 bit extended grayscale image support, image annotation, specialized extended grayscale image display such as window level and LUT processing, and specialized medical image processing.

Other features include lossless JPEG compression, JPIP, and signed and unsigned image data processing.

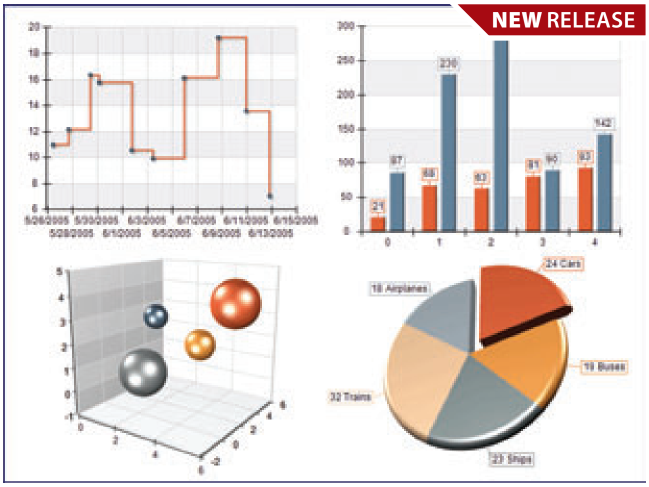
FROM LEAD TECHNOLOGIES

Product	Top 250	Review	Price
LEADTOOLSI magingP roSD K V17.0	57	-	**$895.50**
LEADTOOLS Document Imaging Suite SDK V17.0	-	-	**$4,895.10**
LEADTOOLSMult imediaSD K V17.0	-	-	**$583.10**
LEADTOOLS1D B arcodeMod ule V17.0	-	-	**$660.25**
LEADTOOLSD ocumentI magingSD K V17.0	-	-	**$2,245.50**
LEADTOOLSI CRMod ule V17.0	-	-	**$2,845.25**
LEADTOOLSMe dicalI magingSD K V17.0	-	-	**$4,495.50**
LEADTOOLSOC RMod ule V17.0	-	-	**$1,420.25**
LEADTOOLS VectorI magingP roSD K V17.0	-	-	**$1,795.50**
LEADTOOLS VirtualP rinterSD KMod ule V17.0	-	-	**$1,420.25**

Nevron

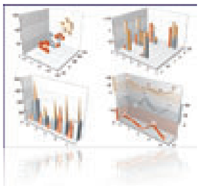
Nevron Chart for .NET Enterprise 2011

A powerful, flexible and extensible charting component for Windows Forms and ASP.NET.



Nevron Chart for .NET (Windows Forms and ASP.NET)

ProfessionalQ uality
Allows you to produce professional presentation quality with stunning visual effects.



Customization and 3D Support
Provides XYZ Scatter Graphs, multiple and controllable axes, scatter stack charts, zooming and scrolling capabilities.

Nevron Financial Charting
Includes charts for specifically targeted software firms: banks, brokerage firms, investment firms and other financial companies.

- Includes many chart types and circular/radial/linear gauges
- Includes scrollable axes, integrated legend, chart annotations, watermarks and an extensible chart layout manager

Nevron Chart for .NET has been designed to be extensible, efficient and feature rich. It can display virtually any 2D or 3D chart including: bar, line, step line, area, pie, point, bubble, stock, floating bar, radar, polar, high low, mesh surface, grid surface, shape, smooth line, floating bar, Venn and error with many variations (like scatter XY and XYZ line, stacked bars etc.).

The component features full Visual Studio design time support with many advanced features like style editors. It also comes with support for many advanced features including: customizable legends, unlimited axes, zooming and scrolling, grouping, filtering, full set of gauge types, advanced formula support, financial and statistical functions and more.

FROM NEVRON

Product	Top 250	Review	Price
Nevron Gauge for Reporting Services	121	-	**$574.28**
Nevron. NET Vision Enterprise2011	141	-	**$1,549.28**
Nevron Chart for .NET Enterprise 2011	145	★★★★	**$866.78**
Nevron Diagram Enterprise for .NET 2011	190	★★★★★	**$866.78**
Nevron Chart for .NET Lite 2011	-	-	**$291.52**
Nevron. NET VisionP rofessional 2011	-	-	**$964.28**
Nevron3D Chart V7.1	-	-	**$671.78**
Nevron Chart for .NET Professional 2011	-	★★★★	**$574.28**
Nevron Chart for Reporting Services	-	-	**$866.78**
Nevron Chart for SharePoint	-	-	**$1,208.03**

version 17

LEADTOOLS®

The World Leader in Imaging SDKs

LEAD TECHNOLOGIES INCORPORATED

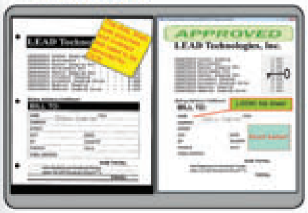
Silverlight, .NET, WPF, WCF, WF, C API, C++ Class Lib, COM & more!

Develop your application with the same robust imaging technologies used by **Microsoft, HP, Sony, Canon, Kodak, GE, Siemens,** the **US Air Force** and **Veterans Affairs Hospitals.**

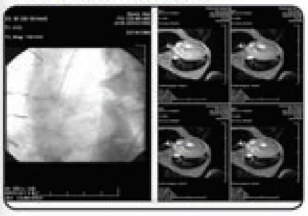
LEADTOOLS provides developers easy access to decades of expertise in color, grayscale, document, medical, vector and multimedia imaging development. Install LEADTOOLS to eliminate months of research and programming time while maintaining high levels of quality, performance and functionality.

- **Silverlight:** 100% pure Silverlight 3 and 4 Imaging SDK.
- **Image Formats & Compression:** Supports 150+ image formats and compressions including TIFF, EXIF, PDF, JPEG2000, JBIG2 and CCITT G3/G4.
- **Display Controls:** ActiveX, COM, Win Forms, Web Forms, WPF and Silverlight.
- **Image Processing:** 200+ filters, transforms, color conversion and drawing functions supporting region of interest and extended grayscale data.
- **OCR/ICR/OMR:** Full page or zonal recognition for multithreaded 32 and 64 bit development with PDF, PDF/A, XPS, DOC, XML and Text output.
- **Forms Recognition & Processing:** Automatically identify and classify forms and extract user filled data.
- **Barcode:** Auto-detect, read and write 1D and 2D barcodes for multithreaded 32 & 64 bit development.
- **Document Cleanup/Preprocessing:** Auto-deskew, despeckle, hole punch, line and border removal, inverted text correction and more for optimum results in OCR and Barcode recognition.
- **PDF & PDF/A:** Read, write and view searchable PDF with text, images, bookmarks and annotations.
- **Annotations:** Interactive UI for document mark-up, redaction and image measurement (including support for DICOM annotations).
- **Medical Web Viewer Framework:** Plug-in enabled framework to quickly build high-quality, full-featured, web-based medical image delivery and viewer applications.
- **PACS Workstation Framework:** Set of .NET PACS components that can be used to build a full featured PACS Workstation application.
- **Medical Image Viewer:** High level display control with built-in tools for image mark-up, window level, measurement, zoom/pan, cine, and LUT manipulation.
- **DICOM:** Full support for all IOD classes and modalities defined in the DICOM standard (including Encapsulated PDF/CDA and Raw Data).
- **PACS Communications:** Full support for DICOM messaging and secure communication enabling quick implementation of any DICOM SCU and SCP services.
- **3D:** Construct 3D volumes from 2D DICOM medical images and visualize with a variety of methods including MIP, MinIP, MRP, VRT and SSD.
- **Scanning:** TWAIN & WIA (32 & 64-bit), auto-detect optimum driver settings for high speed scanning.
- **DVD:** Play, create, convert and burn DVD images.
- **DVR:** Pause, rewind and fast-forward live capture and UDP or TCP/IP streams.
- **MPEG Transport Stream:** With DVR for UDP and TCP/IP streams & auto-live support.
- **Multimedia:** Capture, play, stream and convert MPEG, AVI, WMV, MP4, MP3, OGG, ISO, DVD and more.

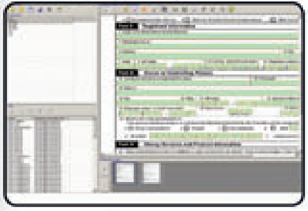
Document



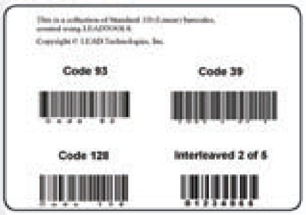
DICOM Medical



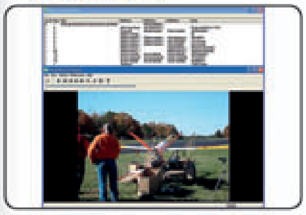
Form Recognition & Processing



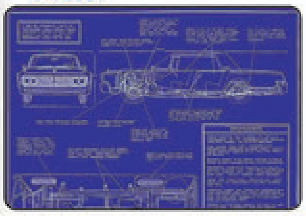
Barcode


This is a collection of Standard 1D (Linear) barcodes, created using LEADTOOLS.
Copyright © LEAD Technologies, Inc.

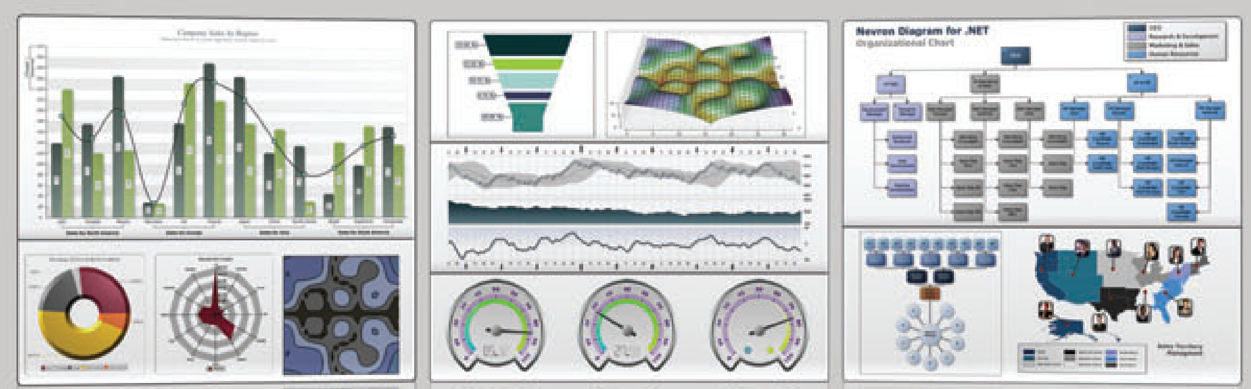
Multimedia



Vector



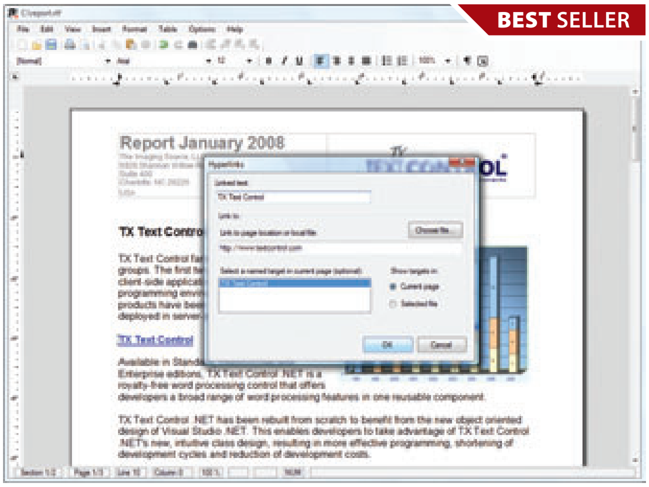
www.componentsource.com Free 60 Day Evaluation!


Nevron Diagram for .NET
Organizational Chart

Text Control

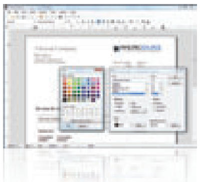
TX Text Control .NET V15.1

Read, edit and create documents in formats like MS Word DOCX, DOC, RTF, PDF, HTML and XML.

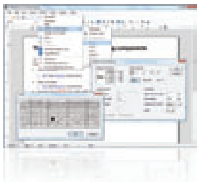


BEST SELLER

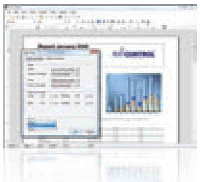
TX Text Control .NET



Character and Paragraph Formatting
Supports both paragraph and character based styles, as well as multiple style inheritance.



Bullets and Numbered Lists
Provides comprehensive support for bullets and numbered lists.



Headers and Footers
End-users can use the same header and footer throughout a document.

- Standardized (ASP.NET & WinForms) WYSIWYG interface
- Supported formats include DOC, DOCX, RTF, TXT, HTML, CSS, XML and PDF
- Create and save ISO conforming PDF/A documents

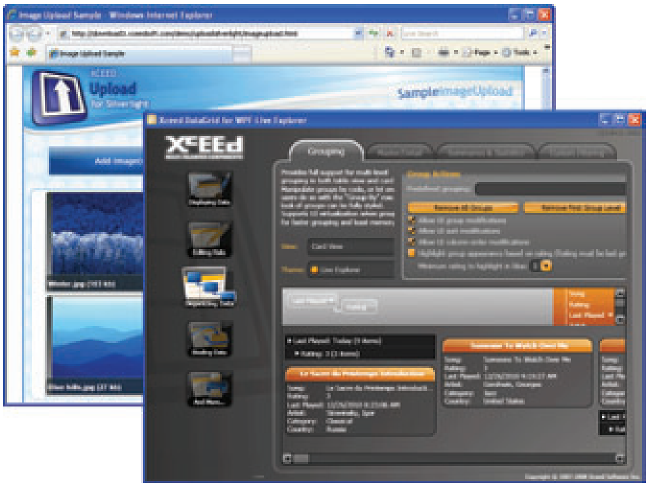
TX Text Control .NET is available in Standard, Professional and Enterprise editions and is a royalty-free word processing control that offers developers a broad range of word processing features in one reusable component. TX Text Control .NET includes the ability to export all documents to print-ready Adobe PDF documents without any third-party software. The fully programmable component supports nested tables, document sections, images, text frames, drag and drop, headers and footers, bulleted and numbered lists and MS Word compatible mail merge fields for database applications. Typical applications include mail merge, PDF creation, document conversion or WYSIWYG word processing.

FROM TEXT CONTROL			
Product	**Top 250**	**Review**	**Price**
TX Text Control .NET Enterprise V15.1	24	-	**$2,182.18**
TX Text Control .NET Server V15.1	37	-	**$2,938.04**
TX Text Control .NET Professional V15.1	58	-	**$1,045.59**
TX Text Control ActiveX Enterprise V15.1	142	-	**$2,182.18**
TX TextC ontrolB undle ProfessionalE dition V15.1	213	-	**$1,714.02**
TX Text Control .NET Standard V15.1	214	★★★★★	**$499.59**
TX TextC ontrolR apidSpell. NET V15.1	-	-	**$293.02**
TX Text Control ActiveX Standard V15.1	-	-	**$499.59**
TX Text Control ActiveX Professional V15.1	-	★★★	**$1,045.59**
TX Text Control ActiveX Server V15.1	-	-	**$2,938.04**

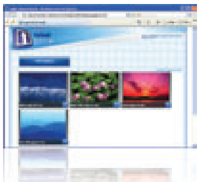
Xceed Software

Xceed Ultimate Suite 2011

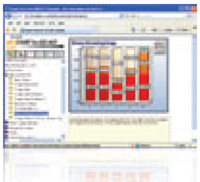
65+ advanced UI controls and data manipulation libraries, for COM, .NET, CF, WF, WPF and Silverlight.



Xceed Ultimate Suite



Xceed Upload for Silverlight
Supports uploading multiple files in a single transaction, and supports GZip and Deflate, with controllable compression.



High-quality, Fast ASP.NET 2D/3D Chart Control
Supports 14 major 2D/3D chart types, flexible axes, scaling, legends, drill-down, etc.

Xceed DataGrid for WPF
Features include hierarchical master-detail; 12 flexible, themed editors; an interactive Cardflow 3D view and more.

- Includes all Xceed's .NET, CF, WPF, WF, Silverlight, COM and ActiveX components
- 1 year Vanguard subscription includes free updates, free upgrades, all new Xceed product releases and priority support

Xceed Ultimate Suite offers both User Interface tools for application front-ends and data handling libraries for application back-ends.

The Suite includes functionality for compression (Silverlight, .NET, ASP.NET, CF, ActiveX), encryption (ActiveX), user interface (WPF,.NET, ASP.NET, ActiveX) data manipulation (WF), file transfer (Silverlight, .NET, ASP.NET, CF, ActiveX) and binary encoding (ActiveX).

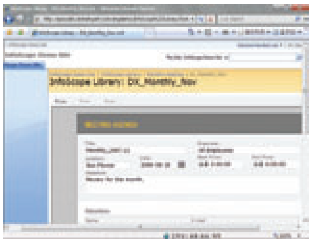
Xceed Grid for .NET (included in Xceed Ultimate Suite) comes with 120+ features: grouping, master/detail, Windows and Office theming, fixed rows and columns, Excel export, gradient maps, interactive visual styles and cell UI virtualization.

FROM XCEED SOFTWARE			
Product	**Top 250**	**Review**	**Price**
XceedF TPf or. NET V5.0	157	-	**$685.95**
XceedU ltimateS uite 2011	161	★★★★	**$1,631.95**
XceedG ridf or. NET V3.9	245	★★★★	**$489.95**
XceedZ ipC ompressionLi brary V6.5	250	-	**$685.95**
XceedZ ipf or. NET V5.0	-	-	**$685.95**
Xceed Zip for .NET with Self-Extractor V5.0	-	-	**$783.95**
Xceed3D Views for WPF V4.0	-	-	**$783.95**
XceedA bsoluteP ackager V1.1	-	-	**$47.95**
XceedC hartf or. NET V4.3	-	-	**$1,175.95**
XceedC hartf orASP.NET V4.3	-	-	**$1,175.95**

.NETXPERT
InfoScope V2.5

Easily post InfoPath forms to a Web server.

- Interoperable with all kinds of .NET based Enterprise solutions
- Includes Windows SharePoint Services (WSS) support



InfoScope

InfoScope is an InfoPath runtime engine that enables users who don't have InfoPath to use InfoPath forms through a Web browser. InfoScope enables inputs and updates of data within InfoPath, and makes the entire process interoperable with a back end system via a Web browser.

InfoScope provides a Service Oriented Architecture (SOA) which meets various different types of complicated business requirements through .NET technology. This enables customers and/or partners to integrate into your business processes via online electronic forms.

InfoScope also provides an InfoScope SharePoint Web Part Control. It's a real InfoPath form not a Web page. No roundtrip and no modification. Just publish and you are ready to use InfoPath. Since InfoScope is auto-installed to clients, you won't meet deployment issues.

InfoScope also enables Web pages to catch events so that you can develop Web applications using InfoPath forms.

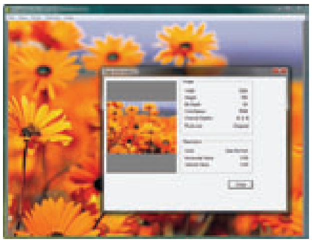
FROM .NETXPERT

Product	Top 250	Review	Price
InfoScope V2.5	-	-	$1,960.00

Accusoft Pegasus
ImageGear for .NET v17

.NET WinForms, ASP.NET and WPF imaging SDK

- Image editing and processing
- Comprehensive OCR capabilities
- Viewing, annotation, and printing



ImageGear for .NET

ImageGear for .NET accelerates .NET imaging application development. Utilizing managed code, it provides support for .NET, ASP.NET, Windows Presentation Foundation (WPF), DirectX 10, and Direct3D 10.

ImageGear for .NET controls the image capture process and strictly adheres to image compression standards. Your users will quickly view over 100 image file types through a common interface including raster files such as TIFF and JPEG, and vector files such as PDF and CAD. The annotation support facilitates enterprise collaboration.

For web applications, ASP.NET offers zero footprint deployments and WPF enhances the end user experience.

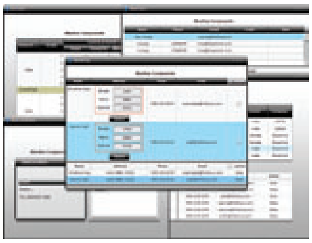
FROM ACCUSOFT PEGASUS

Product	Top 250	Review	Price
FormSuite v2	21	-	$4,899.02
ISISX press V3.0	53	-	$979.02
Prizm Viewer v9	-	-	$93.10
ImageGearf or .NET v17	-	-	$2,939.02
BarcodeX press Standard. NET V6.0	-	-	$1,371.02

ActsOne
BlueOne Components

Add diverse Adobe Flex UI functionality to your Web site.

- Includes grid, grouping, tree, totaling, calculation, input, menus, navigation features and more
- Extends Flex component functions



BlueOne Components

BlueOne Components are custom UI components, developed by ActsOne using Action Script 3.0, with rich functions to make Adobe Flex applications more useful. It showcases many effective, easy-to-use, rich components such as BlueDataGrid, Blue MDI Window and BluePanel. BlueDataGrid in particular has many functions including multi header, data merging, tree datagrid, OLAP grid, Excel export, totaling and sub totaling features. BlueOne Components includes support for Flex based PDF, Word and Flash reporting and has an integrated Web UI as well as an easy-to-use editing environment. Other features include: skin and interface generation, an extended textbox, easy data grid cell extending, performance implementation, automatic cell calculation, cell drill-down functions, column splits, MDI (Multiple Document Interface Window) and an extended tree & button control.

FROM ACTSONE

Product	Top 250	Review	Price
BlueOne Components for Flex Builder 3	-	-	$949.05
BlueOne Components for Flex Builder 2	-	-	$949.05
GridOne V1.0	-	-	$627.00

AG-TECH
Btrieve Classes for .NET V4.0

Fast, easy to program access to Pervasive.SQL databases from .NET.

- Provides direct access to Pervasive's transactional interface from within .NET
- Extended class enables maximum performance when searching Pervasive data

Btrieve Classes for .NET

Btrieve Classes for .NET is a set of .NET database access support classes that enable fast, easy to program access from .NET programs to Pervasive.SQL databases via the Btrieve API. As such they greatly help developers looking to write .NET programs accessing data in Pervasive.SQL databases or those looking to migrate existing pre-.NET Btrieve/Pervasive applications to .NET.

Btrieve Classes for .NET performs with greater efficiency and hence is faster than ADO.NET or Pervasive's OLE DB provider.

This product includes two types of class library, the first (DDF Class) is specifically designed for the .NET Framework and is composed of classes that enable smart and efficient access to column data and definition of data type conversion code. The second type of class (Native Class) provides Btrieve API level access from .NET managed code.

Btrieve Classes for .NET includes full support for Pervasive PSQL v9 and Visual Studio.

FROM AG-TECH

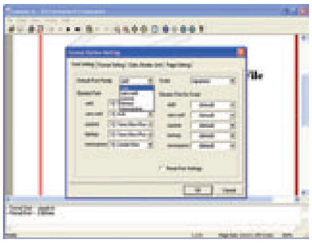
Product	Top 250	Review	Price
BtrieveC lassesf or. NET V4.0	-	-	$764.40

Antenna House

AH XSL Formatter XSL Standard V4.3

Format XML/XSL-FO documents, data and forms at high speed.

- Supports most Unicode languages including Latin, Cyrillic, Greek, Japanese, Chinese, Hangul, Arabic, Hebrew and Thai
- Produces high quality PDF files



AH XSL Formatter

XSL Formatter offers a clear GUI, support for over 50 languages, PDF generation, W3C compliance, extensions and formatting capabilities, unlimited document size and SVG support. In addition XSL Formatter offers vector support for EMF and WMF, CMYK support, and wide ranging output capabilities such as MathML native drawing, WordML transformation, XSL Report Designer integration and PDF enhancement.

It offers the ability to write in vertical mode, to run the text from right to left as in Arabic and Hebrew and also to create multilingual documents with language mixtures on the same page and even on the same line.

FROM ANTENNA HOUSE

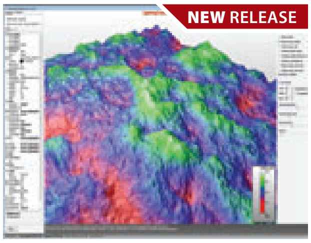
Product	Top 250	Review	Price
AHXSLF ormatter Standard V4.3	196	-	**$4,900.00**
AHC SSF ormatter Lite V5.2	-	-	**$1,960.00**
AHC SSF ormatter Standard V5.2	-	-	**$4,900.00**
AHF ormatter Lite V5.2	-	-	**$2,744.00**
OfficeHTMLFilter V1.1	-	-	**$980.00**

Arction

LightningChart Ultimate 1.3

High-performance data visualization SDK.

- Renders vast data sets instantly
- Detailed high-capacity 3D Surfaces, with sizes over 2000x2000
- Categorizable and customizable legend boxes, with series show/hide functions



NEW RELEASE

LightningChart

LightningChart Ultimate is a high-performance measurement, engineering and research data visualization SDK for .NET. It is an entirely GPU accelerated (Direct3D) and performance optimized data visualization control for presenting masses of data in 2D, 3D and polar views. It is especially designed for professional measurement applications, such as data acquisition, PC-based oscilloscopes, vibration measurement, signal analyzers, scientific research, medical and other real-time measurement and signal monitoring applications. Being optimized for high-speed sampled signal data handling with innovative techniques, it is able to handle incredible data rates.

3D views allow high-quality graphics with many series types. 3D surfaces can be presented with height-coloring, wire-framing, contouring and lighting options.

FROM ARCTION

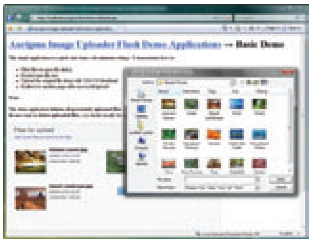
Product	Top 250	Review	Price
LightningChartP ro 1.3	-	★★★★	**$1,783.60**
LightningChart Ultimate1.3	-	-	**$2,420.60**
LightningChartB asic 1.3	-	-	**$382.20**

Aurigma

Aurigma Image Uploader Flash 1.0

Streamline photo uploads to your website with this flash based upload solution.

- Aurigma Image Uploader Flash lets you upload to any server platform
- User-friendly multiple photo uploader
- Resize photos before upload



Aurigma Image Uploader Flash

Aurigma Image Uploader Flash is ideal for Web 2.0 sites with more casual users, it makes photo uploading quick and user-friendly.

Aurigma Image Uploader Flash allows you to upload any file, anywhere. You can easily upload photos, upload videos and upload documents to social network sites, photo sharing sites and content management systems. Just add the Image Uploader code to the upload page on your website, write a few lines of server-side code to save files to a folder or database, and that's it. If you are building your site using PHP or ASP.NET, you can use special wrapper classes which reduce the amount of coding to almost nothing.

FROM AURIGMA

Product	Top 250	Review	Price
Aurigmal mageU ploader Flash1.0	182	-	**$489.02**
AurigmaF ileD ownloader 2.1	-	-	**$146.02**
Aurigmal mageU ploader Java 6.1	-	-	**$146.02**
Aurigmal mageU ploader ActiveX 6.1	-	-	**$146.02**
Aurigmal mageU ploader Dual 6.1	-	-	**$244.02**

BCGSoft

BCGControlBar Library Professional Edition MFC V12.1

Create Office, Visual Studio and IE style user interfaces.

- Includes a customizable Office style Ribbon control
- Includes more than 200 well designed, tested MFC extension classes

BEST SELLER

BCGControlBar Library Professional Edition MFC

BCGControlBar Library Professional Edition MFC has more than 200 (Ribbon, toolbar, menu, calendar, grid, editor, gantt and many others) thoroughly designed, tested and fully documented MFC extension classes which can be easily incorporated into your application to save you hundreds of development and debugging hours. It includes advanced features such as detachable tab panes, auto hide panes, new docking algorithms (similar to those introduced in Microsoft Visual Studio and Microsoft Visio), a text editor with syntax highlighting and IntelliSense-style support, Office-style calendar, professional grids and much more.

BCGControlBar Library Professional MFC includes Source code and a free one year subscription.

FROM BCGSOFT

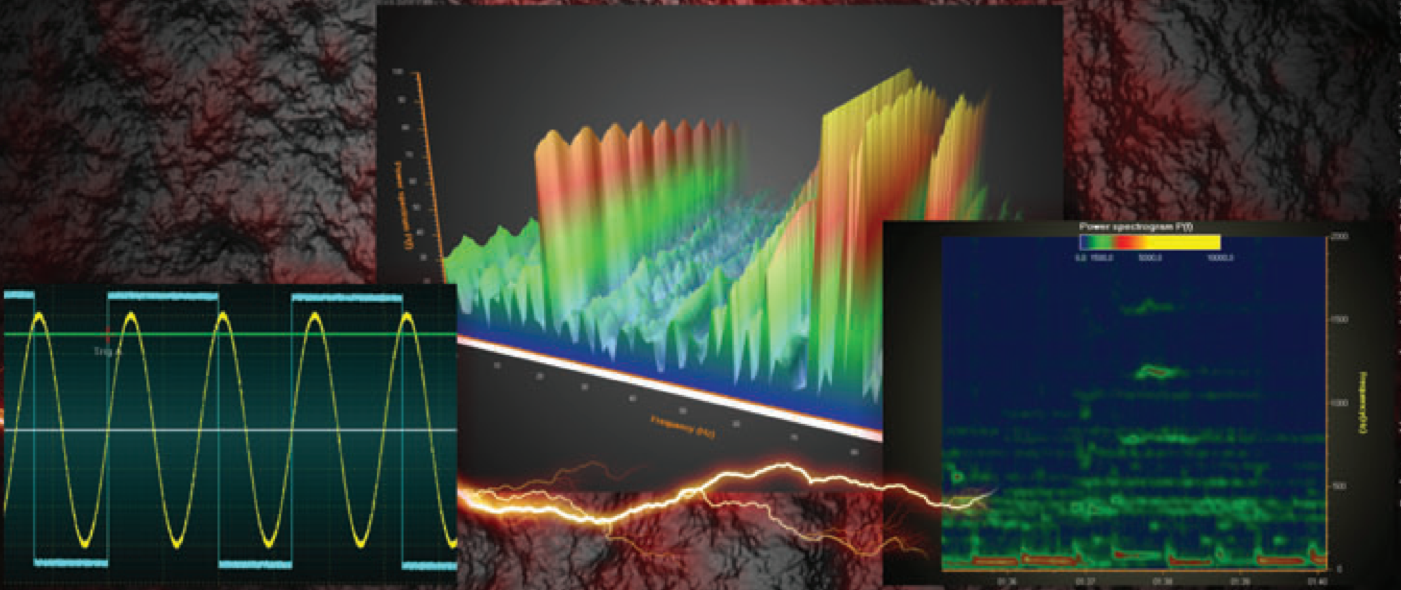
Product	Top 250	Review	Price
BCGControlBar Library Professional Edition MFC V12.1	12	-	**$685.02**
BCGControlBarLi braryf or. NET V4.0	-	-	**$538.02**
BCGPEdit V9.0	-	-	**$195.02**
BCGSuitef orM FC V1.0	-	-	**$489.02**

STUNNING POWER

LightningChart

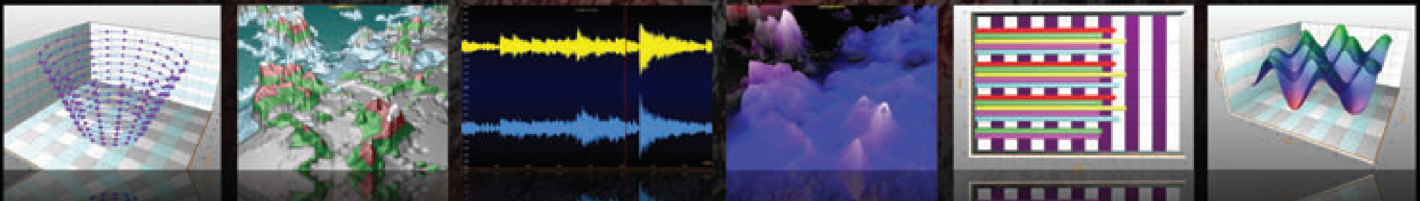
Ultimate
Pro
Basic

The fastest measurement, engineering and research data visualization SDK for .NET



Background image rendered with SurfaceGridSeries3D of LightningChart Ultimate

- Superior 2D and 3D rendering performance - Gigantic 3D surface sizes over 2000x2000 with scrolling features
- Over 100 simultaneous measurement channels, with very high data rates - Most advanced polar charts -



Download a free trial from www.componentsource.com

Instrumentation
Engineering
Science
Automation
Measurement
Analysis
Research
FFT Spectrum
Patient monitors
Data acquisition
Signal monitoring
Vibration measurement
Statistics

Motorsports
Oscilloscopes
Audio waveform editors
Signal analyzers
Seismography
Audio monitoring
Biomedical applications
Automotive industry
Financial applications
Neurophysiology
Audio analysis
Defence technologies
Aerospace

Performance benchmark 1 - Opening large data set

Opening of 1 million points of data, as a line graph. Result is the delay in seconds before data is fully rendered on the screen (lower delay is better).

Chart A	Chart B	Chart C	LightningChart	Chart D	Chart E	Chart F
13	12	5	0.029	13	12.73	9

LightningChart is 37 201% faster than average of others

Performance benchmark 2 - Real-time monitoring

Data is appended to the end of the line series periodically. X axis length is 10 seconds. When 10 seconds elapses, X axis scrolling starts. Test duration is 100 seconds. Result is the point count adding capability / sec, keeping approx. 20 FPS or better refresh rate (higher value is better).

Chart A	Chart B	Chart C	LightningChart	Chart D	Chart E	Chart F
100	100	1000	5000000	10	10	100

LightningChart has 2 272 627% better capacity than average of others

All .NET chart components were configured to equal appearance and size. Benchmark details and source code project are available from Arction Ltd. LightningChart results apply for Ultimate and Pro editions.

Arction

Pioneers of high-performance data visualization

Microsoft Visual Studio

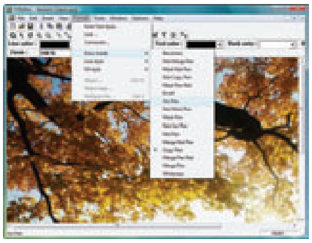
Microsoft .NET

Bennet-Tec Information Systems

MetaDraw ActiveX V3.1

Give your application support for the creation, editing, and displaying of Metafiles.

- Full support for Object Oriented editing: moving objects, sizing, changing colors and other attributes
- Save/load pictures to/from database fields



MetaDraw

MetaDraw is a special purpose picturebox/image editing component. It is an Object Oriented image component for multimedia, image editing/annotation, etc. MetaDraw allows you to create, edit, display and save pictures built from a set of graphic objects. To your application, it looks like a standard picturebox, but it offers unique support for metafiles - a graphical format well suited to the drawing and/or moving of graphical objects - including the ability to tag individual graphic objects for HyperGraphic/HotSpot applications.

You can also use MetaDraw as a metafile viewer with support for zoom, scroll and even selective display of individual elements.

FROM BENNET-TEC INFORMATION SYSTEMS

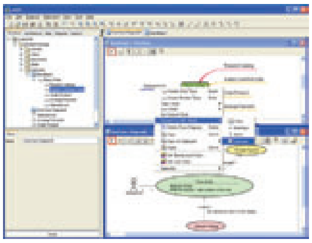
Product	Top 250	Review	Price
MetaDraw ActiveX V3.1	235	★★★	$479.04
ALLTextHT /Pro OCX V4.5	-	-	$343.00
MetaDraw WinForms for .Net	-	-	$489.02
TBackP ro V2.0	-	-	$147.00
Tlist V8.0	-	-	$383.04

Change Vision

astah* professional

Business process UML modeling tool.

- Edit model information directly, without opening dialog boxes
- Supported diagram types include Class, UseCase, Sequence, Collaboration, StateChart, Activity, Component and Deployment



astah* professional

astah* professional is a system design tool which supports UML, Entity Relationship Diagrams, flowcharts, CRUD, data flow diagrams, requirements tables and Mind Maps.

astah* professional includes basic features to make modeling simple and user friendly. astah* professional is suitable for business use, handling large-sized models and document creation. It includes enriched features, such as the ability to change your view freely using a map view, zoom by dragging, unlimited undo/redo, the ability to output project information to a CSV file, automatic generation of class diagrams with model information, diagram creation guidance and more.

Exporting to HTML Documents (Javadoc) is also available and enables people who don't have astah* professional to see models by using a browser.

FROM CHANGE VISION

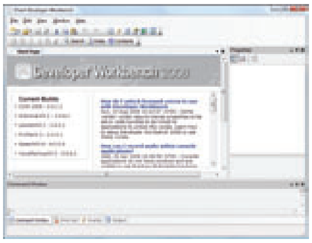
Product	Top 250	Review	Price
astah*p rofessional V6.2	127	★★★★★	$274.40
astah*U ML V6.2	-	-	$49.00
astah*s hare V2.2	-	-	$686.00

Chant

Chant Developer Workbench

Comprehensive development and testing environment for working with speech.

- ActiveX, C DLL, COM, Java, .NET, VCL, Web, Win32, Win64 & WinCE libraries
- Supports Dragon, ViaVoice, SAPI, and VoCon recognizers
- Supports Acapela, Cepstral, RealSpeak Solo, & SAPI synthesizers



Chant Developer Workbench

Chant Developer Workbench is comprised of tools and components for developing software that speaks and listens. As an interactive toolset, it provides a development and testing environment for working with the component libraries and the speech technology objects they manage. It includes a multi-document, interactive, customizable environment, a powerful editor with color-coded formatting, command line testing and event tracing.

With the component libraries, you can manage grammars, lexicons, profiles, recognizers, synthesizers, and text-to-speech markup directly within application software you develop and deploy.

FROM CHANT

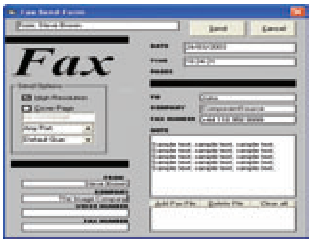
Product	Top 250	Review	Price
ChantD eveloper Workbench	-	-	$881.02
ChantG rammarKit V3.0	-	-	$195.02
ChantL exiconKit V2.0	-	-	$195.02
ChantP rofileKit V3.0	-	-	$195.02
ChantS peechKit V6.0	-	-	$195.02

Data Techniques

FaxMan V4.6

Seamlessly embed fax support into your Windows applications.

- Create faxes from any application using the FaxMan printer driver
- Sophisticated logging keeps track of every fax you schedule
- Supports almost all Class 1, Class 2, and Class 2.0 fax modems



FaxMan

FaxMan is a programmable fax engine for providing fax sending and receiving capability with total control over the faxing process and the user interface. The developer kit includes ActiveX OCX, DLL and managed .NET component interfaces to support virtually any Windows application, and automatically works with almost all Class 1, Class 2, and Class 2.0 fax modems. FaxMan includes sample source code in Visual Basic and Visual C++.

The FaxMan SDK allows developers to integrate Fax support for industry standard fax modems in hours instead of the months it would take to write support from scratch.

FROM DATA TECHNIQUES

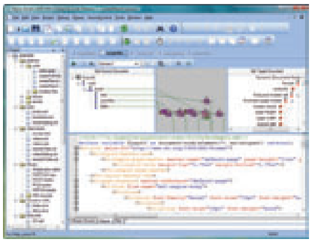
Product	Top 250	Review	Price
FaxMan V4.6	-	-	$763.20
FaxManJ r. V2.20	-	★★★★★	$475.20
FaxMan PDF Import Addon	-	-	$583.10
ImageIt V1.1	-	-	$78.35
ImageMan ActiveX V8.72	-	★★★★★	$485.10

DataDirect Technologies

Stylus Studio® XML Professional Suite 2011

Comprehensive XML data transformation and aggregation toolset.

- Includes synchronized visual XML editing views, Sense:X, integrated XML validator, XML differencing, and more
- Supports visual XQuery editing, mapping and debugging



Stylus Studio® XML

Stylus Studio XML Enterprise Suite is an advanced XML integrated development environment (XML IDE) consisting of XML tools and high-performance Java and .NET components for deploying data integration applications.

Stylus Studio XML Enterprise Suite includes features for working with XML, XQuery, XML Pipeline, XSLT, XSL:FO, EDI, XML Schema/DTD, XPath, XML & databases, XHTML, XML mapping, XML publishing and Web services.

Stylus Studio XML also features Electronic Data Interchange (EDI) integration tools for converting (EDIFACT to XML, X12 to XML), mapping and deploying advanced XML data integration applications which involve read/write access to legacy EDIFACT and X12 data sources. It also lets you create XML schema files that correspond to the converted EDI documents.

FROM DATADIRECT TECHNOLOGIES

Product	Top 250	Review	Price
StylusS tudio®XML ProfessionalS uite 2011	106	-	$441.00
StylusS tudio®XML EnterpriseS uite 2011	163	-	$681.10
StylusS tudio®XML AcademicE nterprise 2011	-	-	$392.00

DBI Technologies

Solutions Schedule for .NET V3.1

Build team management and resource planning solutions.

- Share schedule, appointment, contact, location and task information
- Supports timebar linking, cell-level formatting, multi-select timebar and more

Solutions Schedule for .NET

Solutions Schedule for .NET allows developers to create scheduling solutions such as those found in the healthcare or services sector. Solutions Schedule for .NET includes intuitive mouse driven scheduling and planning with Gantt-style project management features. Other features include independent cell formatting, timebar overlap, split and linking styles. You can easily add a visual reference point for the current date and time in a schedule by using the Current Time Line feature. Time lines extend vertically through the schedule area and are drawn along the major time segments. Developers can also add zoom-in/out capabilities by programmatically changing the value of the TimeDistance property.

FROM DBI TECHNOLOGIES

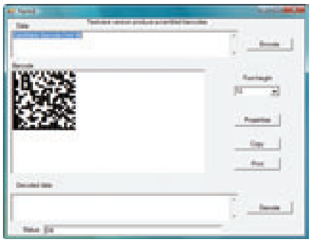
Product	Top 250	Review	Price
SolutionsS chedulef or. NET V3.1	51	-	$969.22
StudioC ontrolsf or. NET V1.1	-	-	$383.04
StudioC ontrolsf orC OM V1.1	-	-	$719.04
DBIS taff-Scheduler 3.2	-	-	$773.22
Extractor V7.2	-	-	$7,350.00

dLSoft

DataMatrix Font Kit V5.10

Create DataMatrix barcodes from within your applications.

- Includes TrueType, OpenType and PostScript fonts
- Easily translate data into the text required to produce the barcode in the supplied font



DataMatrix Font Kit

DataMatrix Font Kit provides a versatile way to create DataMatrix barcodes from within any application written in a programming language that supports accessing a DLL, OCX, .NET Class, Java Class or UFL (User Function Library). DataMatrix Font Kit translates data into a form which may be displayed or printed in the barcode fonts, which are provided as TrueType, OpenType and PostScript fonts in three different weights and five height/width ratios. The component can return the string, copy it to the clipboard, or write it to a named file.

DataMatrix Font Kit includes samples for MS Office, VB, VB.NET, C#, Crystal Reports and Delphi. (Crystal Reports barcodes are limited to approximately 100 characters).

FROM DLSOFT

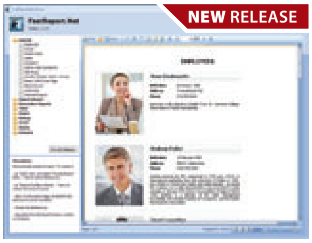
Product	Top 250	Review	Price
DataMatrixF ontK it V5.10	246	-	$253.29
QRC odeF ontK it V5.10	-	-	$253.29
dBarcode.NETS tandard V4.0	-	-	$202.75
dFont Barcode Fonts for Windows - Code 128/EAN 128 V5.0	-	-	$205.50
ReallyS impleB arcodes V3.20	-	-	$125.85

Fast Reports

FastReport.Net 1.5

Create application independent reports.

- Wide range of report objects available: text, picture, lines, shapes, barcodes, table, checkbox, chart, etc
- You get a full-featured reporting solution with a runtime royalty-free report engine and designer



FastReport.Net

FastReport.Net is a full-featured reporting solution for Windows Forms and ASP.NET that can be used in Visual Studio 2005/2008/2010. With FastReport.Net, you can create application-independent reports or FastReport.Net can be used as a standalone reporting tool. You can connect to any database, use any table or create and use a query to form your reports. FastReport.Net lets you add dialogue form(s) to your report and using built-in scripting, you can manage interaction between a form's dialogue controls and perform complex data handling. You can view the resulting reports, print them and export to many common document formats.

FROM FAST REPORTS

Product	Top 250	Review	Price
FastReport.Net 1.5	201	-	$293.02
FastReportP rofessional V4.10	-	-	$244.02
FastCube 1.8	-	-	$175.42
FastQueryBuilder 1.03	-	-	$67.62
FastReport Basic V4.10	-	-	$97.02

FOSS Software

Prof-UIS V2.90

Create professional, modern user interfaces for your Windows applications.

- Microsoft Office and Visual Studio look-and-feel
- Includes tabbed docking windows, tab containers, data grid and property grid controls

Prof-UIS

Prof-UIS is an easy-to-use MFC extension library that enables you to deliver Windows applications with a professional and user-friendly interface. More than 200 thoroughly tested and documented C++ Classes, templates and interfaces will help you take the time and complexity out of incorporating rich and up-to-date GUI facilities in your projects including: Office and Visual Studio look-and-feel, ribbons, tabbed docking windows, tab containers with detachable tabs, customizable data grid and property grid controls, customizable menus, toolbars and keyboard accelerators, multi profile GUI persistence and more. Prof-UIS is an extensible, object-oriented framework that enables you to easily customize and extend its functionality.

FROM FOSS SOFTWARE

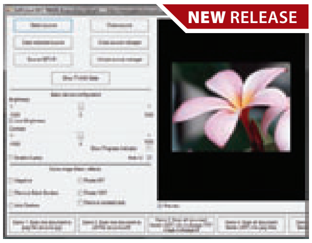
Product	Top 250	Review	Price
Prof-UIS V2.90	216	-	$338.10
ElegantR ibbon V3.7	-	-	$142.10
ElegantG rid V1.3	-	-	$93.10
Prof-UISF rame Features V1.2	-	-	$191.10

GdPicture

GdPicture.NET V7.2

Add advanced royalty free image processing to your applications.

- Display Raster Images, Metafiles & PDFs within a full-featured document viewer
- Full featured & royalty free document imaging toolkit delivered as .NET components



GdPicture.NET

GdPicture.NET is a suite of document imaging toolkits delivered as .NET components enabling developers to compose, display, capture, edit and print documents in .NET applications. With GdPicture.NET your programs will be able to display documents, acquire images from TWAIN scanners, perform optical character recognition (OCR) and many other features covering all mainstream areas of document imaging.

GdPicture.NET technology supports more than 45 vector and bitmap image formats from 1-bit to 128-bit including High Dynamic Range images, JPEG2000, JBIG, PSD, JPEG, TIFF, PNG and more.

FROM GDPICTURE

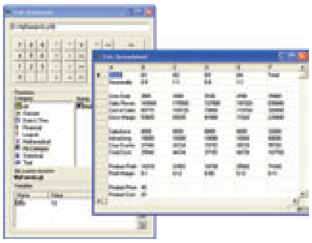
Product	Top 250	Review	Price
GdPicture.NET V7.2.0	60	-	$2,351.02
GdPictureP rol magingSD K V5.13	90	-	$734.02
GdPicture TesseractP lugin	-	-	$489.02
GdTwainP ro SDK V2.8	-	-	$538.02
GdViewerP roA ctiveX V4.12	-	-	$489.02

Inabyte

InaCalc.Net V3.02

Enable advanced calculations in your accounting or spreadsheet application.

- Integrates expression parsing, evaluating, and dependent formula recalculation
- Includes custom function support and multiple formula recalculation



InaCalc.Net

InaCalc.Net supports conventional math operators and functions and is suitable for heavy-duty number crunching. It also supports other data types, such as string, date-time and logical and comes with operators and functions for these data types. It allows defining variables and sets of related formulas that implement spreadsheet-like recalculations. The InaCalc.Net component integrates expression parsing, evaluating and dependent formula recalculation. It is useful in two general areas; when a formula has to be defined, and, evaluated at runtime.

The InaCalc.Net component also supports custom functions and virtual variables.

FROM INABYTE

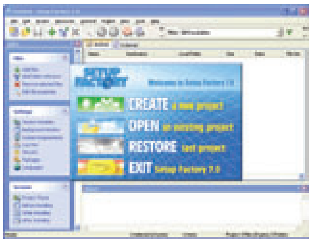
Product	Top 250	Review	Price
InaCalc.Net V3.02	-	-	$391.02
InaAuthenticate V2.1	-	-	$273.42
InaAuthenticate.Net V3.2	-	-	$322.67
InaCalc V2.5	-	-	$273.42
InaCardCheck V3.0	-	-	$230.30

Indigo Rose

Setup Factory V8.2

Create professional installations using a clear visual design environment.

- Wizard based creation of installation packages (no coding needed)
- Code editor for advanced developers to 'fine tune' installations
- Extensive customization through actions library, themes and skins

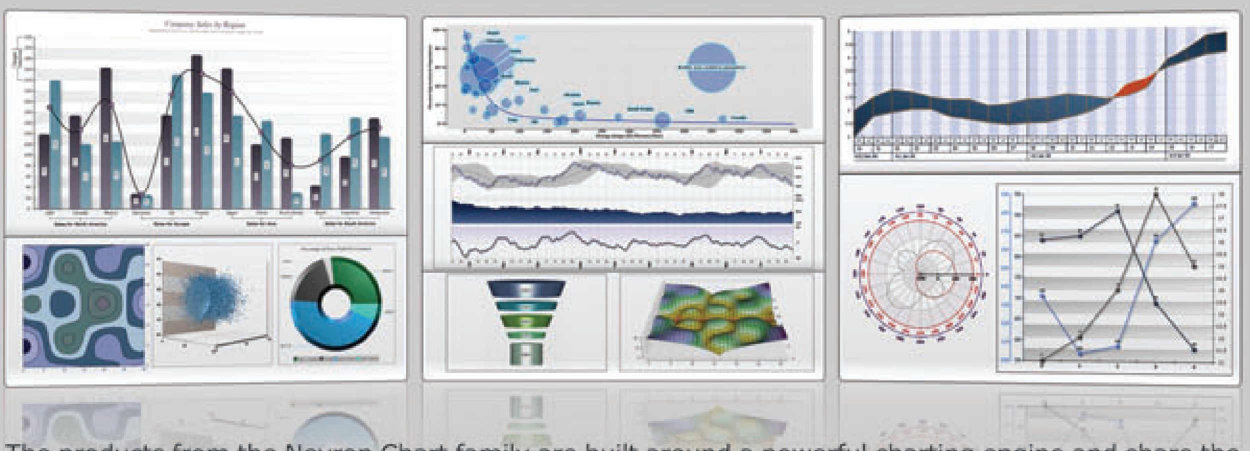


Setup Factory

Setup Factory offers developers a solution for creating flexible installation systems without needing to learn a proprietary scripting language. It further sets itself apart by providing the customization and advanced control features that developers require, all from within the Setup Factory Visual Design Environment. You can even deploy your applications on Web servers. Setup Factory contains a scripting engine that offers all the usual programming logic such as loops, functions and arrays and more. It includes an action wizard that makes it easy, even for someone with zero programming knowledge to simply click to add actions. Also included is a script editor that makes it easy for advanced users to edit code directly.

FROM INDIGO ROSE

Product	Top 250	Review	Price
SetupF actory V8.2	166	★★★★	$387.10
AutoPlayMe diaS tudio V8.0	177	★★★★★	$289.10
TrueUpdate V3.5	-	-	$387.10
MSIF actory V2.1	-	-	$387.10
VisualP atch V3.5	-	-	$387.10

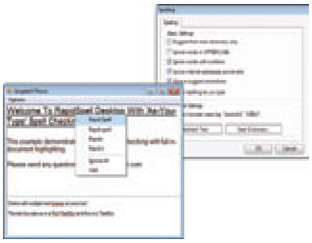


Keyoti

RapidSpell Desktop .NET V4.7

Integrate spell checking functionality into your .NET and WPF applications.

- Interactive dialog checking
- Includes support for 3rd party components
- WPF support includes dialog based checking of WPF's TextBox and RichTextBox controls



RapidSpell Desktop .NET

RapidSpell Desktop .NET allows you to add "as-you-type" spell checking to any textual application (email, word processor etc.) with a couple of lines of code. It can also be used in non GUI contexts using its spell checking API.

RapidSpell Desktop .NET provides 3 spelling components; RapidSpellDialog (dialog based checker control), RapidSpellAsYouType (as you type check control) and RapidSpellChecker (core spell checker class).

RapidSpell Desktop .NET includes 140,000 word U.S. & U.K. dictionaries (and user dictionary), multi-threaded GUI and a suggestion engine. Additional dictionaries are available for other languages.

FROM KEYOTI

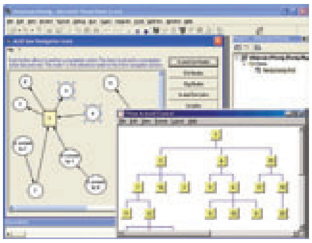
Product	Top 250	Review	Price
RapidSpellD esktop. NET V4.7	-	★★★★★	$465.50
Keyoti Search Lite ASP.NET 2010.1	-	-	$425.32
Keyoti Search Pro for ASP.NET 2010.1	-	-	$686.00
RapidSpellJa va Desktop V2.2.1	-	-	$224.42
RapidSpellJa va Web V3.1	-	-	$195.02

Lassalle Technologies

AddFlow Suite V5.4

Flowchart/diagram creation and graph drawing algorithms.

- Link autorouting, gradient colors
- OwnerDraw events, custom shapes
- Links can be composed of many segments



AddFlow Suite

AddFlow Suite is a suite of components that includes AddFlow and LayoutFlow, a collection of 4 layout components: HFlow (hierarchical), SFlow (symmetric), TFlow (tree) and OFlow (orthogonal). It lets you create interactive flowcharts and workflow diagrams with graph drawing algorithms. Drawings can be made interactively or programmatically and features include distinct shapes, styles, pictures, colors, fonts, user data, and so on for each object of the diagram. Metafile support, XML serialization, printing, zooming, custom shapes, link autorouting, scrolling, clipboard management, grid support, multi-selection and customization are also supported.

FROM LASSALLE TECHNOLOGIES

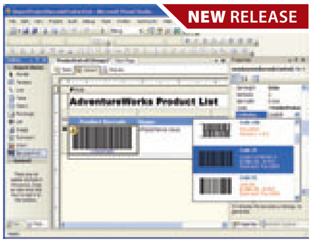
Product	Top 250	Review	Price
AddFlowS uite V5.4	109	-	$979.02
AddFlow ActiveX V5.4	150	★★★★	$489.02
AddFlowf or. NET V2.3	194	-	$489.02
AddFlowS uitef or .NET V2.2	-	-	$685.02
AddFlowf orS ilverlight Professional V1.3	-	-	$685.02

Neodynamic

Neodynamic Barcode Professional for RS V7.0

Add barcode images to your Reporting Services reports.

- Barcode image generation and printing support for 53 Linear/1D & 9 2D Postal barcode symbologies
- Automatically computes checksum or check digits for all symbologies



Neodynamic Barcode Professional for RS

Barcode Professional for Reporting Services is a .NET component which allows you to add barcode images to Reporting Services reports. Barcode Professional for SSRS can be used with both Reporting Services 2000 and 2005 and can generate most popular Linear 1D and 2D barcode symbologies including USPS, Royal Mail, Australia Post, Codabar, Code 11, Code 16k, Code 39, Code 93, Code 128 (A, B, C), Data Matrix ECC200, EAN 8, EAN 13, JAN, Industrial 2 of 5, Interleaved 2 of 5, ISBN, PDF 417, Postnet, UCC EAN 128, UPC A, UPC E and many more.

Barcode images can be generated in formats such as GIF, JPG/JPEG, BMP, and PNG. You can rotate the barcode image in 90, 180 and 270 degrees.

FROM NEODYNAMIC

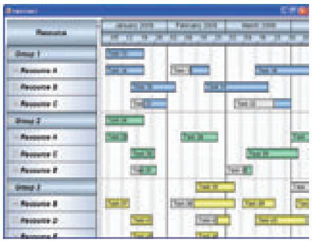
Product	Top 250	Review	Price
Neodynamic Barcode Professional for Reporting Services Ultimate V7.0	135	-	$705.40
Neodynamic Barcode Professional for Windows Forms Standard V7.0	218	-	$293.90
ThermalLabel SDK for .NET 3.0	221	-	$340.92
Neodynamic Barcode Professional for ASP.NET Standard V7.0	-	-	$293.90
ImageDrawSD Kf or. NET V3.0	-	-	$274.30

NETRONIC Software

VARCHART JGantt V2.4

Add interactive Gantt diagrams to your applications.

- User interactions such as create, delete and drag & drop activities are handled by the component
- Includes zooming and scaling of diagrams



VARCHART JGantt

VARCHART JGantt is a component written in Java, that easily lets you integrate Gantt charts into your application. Activities in VARCHART JGantt can be displayed either by a variety of pre-defined bar shapes, or can be composed from scratch. Any shape and pattern is available, even dynamic bars that actively grow or diminish to indicate the degree of completion of an activity. Bars can also be connected by links of different types. Vertical and horizontal grids can be placed in the Gantt graph and the time scale can be customized by position, scaling, colors and font.

Settings to define the appearance and behavior of the Gantt chart can be defined at design time.

FROM NETRONIC SOFTWARE

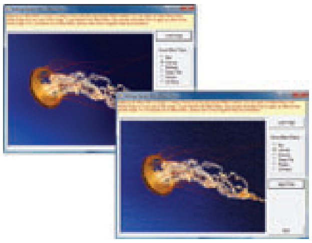
Product	Top 250	Review	Price
VARCHARTJG antt V2.4.4	-	-	$2,909.62
VARCHARTX Gantt .NET V4.4	-	-	$2,674.42
VARCHARTX GanttA ctiveX V4.4	-	-	$2,674.42
VARCHARTXN et .NET V4.3	-	-	$2,841.02
VARCHARTX Tree. NET V4.3	-	-	$2,135.42

Newtone

ImageKit.NET V2.0

100% native .NET image processing component.

- Load and save images to a variety of image formats
- Apply various imaging effects and transformations and display those images in various controls



ImageKit.NET

ImageKit.NET lets you add image processing to your .NET applications quickly and easily. You can retrieve images from TWAIN scanners and digital cameras; load & save image files in a variety of formats; apply image filters and transformations to your images; display images in the display, pan window or thumbnail controls.

ImageKit.NET supports a variety of raster and vector file formats and compressions. Native .NET support includes: BMP, JPEG, GIF, TIFF, PNG, WMF and EMF. ImageKit.NET includes many image effects and transformations including: creating or copying images, combining images, layering images, copy and paste images plus numerous effect filters including blur, mosaic, outline, emboss, oilpaint, canvas, glasstile, lens, ripple, whirlpinch, waves and motionblur. ImageKit.NET contains components for both Windows Forms and Web Forms applications.

FROM NEWTONE

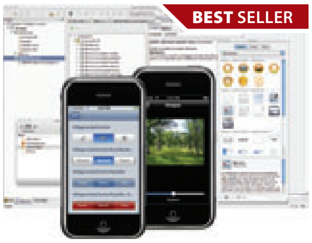
Product	Top 250	Review	Price
ImageKit.NET V2.0	-	-	$704.72
ImageKit7	-	★★★★★	$391.02
ResizeKit V2.0	-	-	$87.32

Novell

MonoTouch 3.0

Create C# and .NET applications for iPhone, iPod Touch and iPad.

- Distribute your applicaions on the Apple App Store
- Turn .NET executables and libraries into native applications



MonoTouch

MonoTouch from Novell is a software development kit that contains a suite of compilers, libraries and tools for integrating with Apple's iPhone SDK. MonoTouch includes Microsoft .NET base class libraries along with managed libraries for taking advantage of native iPhone APIs. Also included is a cross-compiler that can be used for turning .NET executable files and libraries directly into native applications for distribution on the Apple Apps Store or for deployment to enterprise iPhone users. In addition, Xcode integration enables application developers to test on the device or in Apple's iPhone Simulator.

MonoTouch is an SDK that can be used with your favorite editor, be it a fully integrated development environment or a simple text editor. The MonoDevelop integration helps developers get started by providing iPhone application templates that will get you from zero to your device in no time.

FROM NOVELL

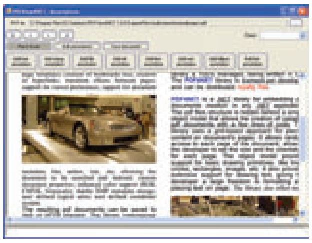
Product	Top 250	Review	Price
MonoTouch 3.0	48	-	$391.02
Mono Toolsf or VisualS tudio 2.0	110	-	$97.02

O2 Solutions

PDFView4NET V3.0

Add PDF viewing to your .NET applications.

- Supports the displaying and printing of PDF files
- Includes a separate library for converting PDF files to images and for printing PDF files to any Windows printer without any UI



PDFView4NET

PDFView4NET is a .NET toolkit for displaying and printing PDF files in .NET applications. It includes a PDF viewer control for Windows Forms and a .NET library for rendering and printing PDF files from any .NET application.

The PDF viewer control for Windows Forms includes support for annotating PDF files, bookmarks navigation, adding and removing file attachments and other features. The toolkit includes its own PDF rendering engine and it does not rely on any other software for rendering and printing PDF files.

PDFView4NET toolkit has been developed entirely in C#, being 100% managed code.

FROM O2 SOLUTIONS

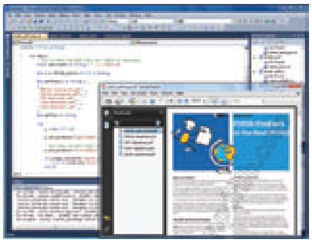
Product	Top 250	Review	Price
PDFView4NET V3.0	99	★★★★★	$685.02
PDF4NET V4.1.3	208	★★★★	$489.02
Barcode4NET .NET V2.0	-	-	$489.02
Barcode4NET WindowsF orms&Mob ileD evices V2.0	-	-	$293.02
Barcode4NET ASP.NET&Re portingS ervices V2.0	-	-	$293.02

PDFlib

PDFlib 8.0

Create PDF files from within server-side or client-side software.

- Includes functions for creating text, graphics, images and links
- Dynamically generate PDF files from Web or database server data
- New table formatter



PDFlib

PDFlib is a component that allows you to programmatically create PDF files from within your own server-side or client-side software. PDFlib doesn't make use of third-party software for generating PDFs, nor does it require any other tools. PDFlib is available for all major operating systems and development environments. It offers a variety of functions, including a table formatter, PDF/A output for longtime archiving, AES encryption, integrated pCOS analyzing tools, a repair and optimization mode for inputing PDFs.

PDFlib can generate PDF data directly in memory (instead of on file), resulting in better performance and avoiding the need for temporary files which speeds up PDF generation.

FROM PDFLIB

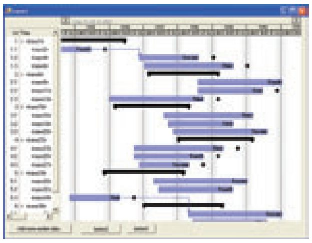
Product	Top 250	Review	Price
PDFlib 8.0	-	-	$1,070.34
PDFlibp COS V2.0	-	-	$262.54
PDFlib Personalization Server (PPS) 8.0	-	-	$3,217.75
PDFlibP LOP DS V4.0	-	-	$219.59
PDFlib TET V4.0	-	-	$407.41

plexityHide.com

GTP.NET V3.3

Add interactive Gantt charts and scheduling to your applications.

- Supports AJAX with client-movable and resizable time items
- Features asynchronous collision detection and handling of time items
- Security awareness enables deployment on XBAP or ClickOnce



GTP.NET

GTP.NET's overall goal and purpose is to let you visualize and manipulate time based information. GTP.NET includes a collection of user interface components which handle time blocks. The time blocks are displayed in a Gantt chart or a schema view. The times are connected to an owner, for example a person or an activity in a project. You can also use it to display dependencies and collisions between times, which gives the end user a graphical way to view and modify the project.

GTP.NET includes the ability to select and re-assign links and also provides databinding for links between time items. The components can be delivered via WinForms, AJAX, ASP.NET, and even in reduced trust environments like ClickOnce and XBAP.

FROM PLEXITYHIDE.COM

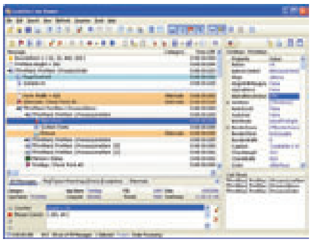
GTP.NET V3.3	74	★★★★	$955.20
Gantt TimeP ackage VCLE dition V3.x	191	-	$469.44
Gantt TimeP ackageA ctiveXE dition V3.x	-	★★★★	$469.44
GTP.NETf orS ilverlight + WPF V1.1	-	-	$1,225.00

Raize Software

CodeSite V4.5

Locate and eliminate problems in your Delphi and C++Builder code.

- Captures detailed information
- CodeSite messages do not interrupt the flow of your application
- Live logging and file logging can be performed either locally or remotely



CodeSite

CodeSite is an add-in for Delphi and C++Builder that enables you to send detailed information (including string lists, objects, even bitmaps) from within your application code to a specialized receiver. Unlike message boxes and inspecting variables on breakpoints, CodeSite messages are not transient. The resulting log of messages sent to a destination provides valuable information for locating problem areas in your code. Sending CodeSite messages does not interrupt the flow of your application as happens when message boxes and breakpoints are used. As a result, CodeSite is much more effective in situations where user interactions (e.g. focus changes) and painting issues need to be tracked.

FROM RAIZE SOFTWARE

Product	Top 250	Review	Price
CodeSite V4.5	-	-	$391.02
DropMaster V2.1	-	-	$97.02
Inspex V2.3	-	-	$97.02
Raize Components V5.2	-	★★★★★	$391.02
ScratchPad V2.0	-	-	$28.42

Resco

Resco MobileForms Toolkit 2011 Volume 1

Suite of .NET CF controls for Pocket PC, Smartphone & Windows CE.NET.

- Includes grid, calendar, listview, input form, zip compression, navigation and touchscreen signature controls
- Features a chart control with five different chart types

Resco MobileForms Toolkit

Resco MobileForms Toolkit is a suite of components designed specifically for mobile devices. It includes advanced designers, support of quick data loading, standardized mobile user interface, Enterprise samples and many other features. It contains AdvancedList, AdvancedTree, CompactChart, DetailView, SmartGrid, Zip and Outlook Controls.

All Resco MobileForms Toolkit components have been specifically designed so they fit the small displays of mobile devices perfectly.

Special optimizations such as Resco DelayLoad technology ensures performance is optimized even on devices with limited memory and processor capabilities.

FROM RESCO

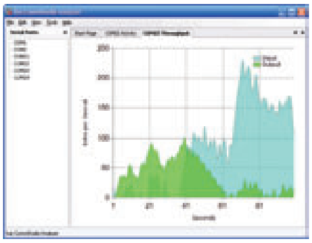
Product	Top 250	Review	Price
RescoMob ileForms Toolkit 2011 Volume1	43	-	$783.95
RescoMob ileAppS tudio 2	189	-	$1,175.95
RescoA ssetInventoryE SP	-	-	$3,429.95
Resco MobileBusiness ESP	-	-	$3,429.95
RescoMob ileSurveyE SP	-	-	$3,429.95

Sax.net

CommStudio

Serial Communications made easy.

- Supports common file transfer protocols, serial ports and modems
- Includes ActiveX components compatible with the MSComm control



CommStudio

CommStudio makes it easy to let your application communicate using the serial port and modems. Sax CommStudio provides a complete communications solution for both COM and .NET. The tools integrate remote systems and devices with Visual Studio .NET and Visual Basic 6. CommStudio's robust communications components based on more than ten years of market leadership are only the beginning, the real gain in developer productivity comes from Sax CommStudio's advanced debugging and analyzing tools. Sax CommStudio supports a large variety of serial port connections including: standard built-in serial ports, virtual serial ports on other systems via network software, Bluetooth and infrared serial ports, high performance serial ports using specialized hardware such as Digiboards etc. The number of serial ports supported on a single system is only limited by hardware.

FROM SAX.NET

Product	Top 250	Review	Price
CommStudioA ctiveX	-	-	$949.05
CommStudio. NET	-	-	$783.02
CrescentQ uickPak V4.5	-	-	$759.05

rssbus

ADO.NET PROVIDERS for **Google & QuickBooks**

- Build cutting edge .NET applications that connect to any data source with ease.
- Easily "databind" to applications, databases, and services using standard wizards.
- Comprehensive support for CRUD
- ADO.NET Data Providers integrated with Visual Studio

Google

Google Data Provider

Integrate .NET applications with Google Apps

ADO.NET access to GMail, Google Docs, Search, Adwords, Google Calendar, Google Talk, and more!

QuickBooks

QuickBooks Data Provider

Integrate .NET applications with QuickBooks

ADO.NET access to QuickBooks customers, transactions, invoices, sales receipts, reports, and more!

DataBind to

SharePoint
QuickBooks
Anything...
Google
Salesforce

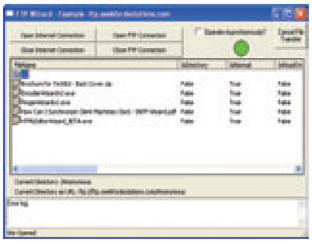
New Data Providers Coming Soon!
For a free trial visit: **www.componentsource.com**

Seekford Solutions

FTP Wizard V3.0

Add FTP support to your Windows applications.

- Add functionality without learning the complexities of FTP or TCP/IP
- Includes samples for Visual Basic, Visual FoxPro, Visual C++, ASP and VBScript



FTP Wizard

Seekford Solutions FTP Wizard allows you to easily upload and download files from an FTP server. Using the File Transfer Protocol, you can get directory listings, make directories, delete files and directories, rename files, upload and download files. The product makes implementation simple. It provides an interface to the File Transfer Protocol, allowing developers to implement this functionality without learning the complexities of FTP and TCP/IP. Multiple instances of FTP Wizard can be created to talk to multiple FTP sites at a time.

FTP Wizard includes asynchronous operation and allows you to create applications that require no user interaction.

FROM SEEKFORD SOLUTIONS

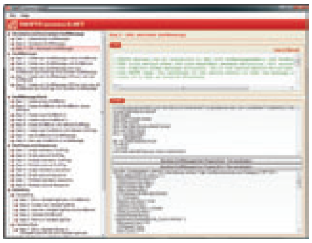
Product	Top 250	Review	Price
FTP Wizard V3.0	-	★★★★★	$191.04
SMTP Wizard SSL V3.0	-	-	$191.04
.NETD NSMX Wizard V3.0	-	-	$195.02
.NETE mailValidation Wizard V3.0	-	-	$195.02
.NETE ncoder Wizard V3.0	-	-	$195.02

SWIFT Components

SWIFT Framework .NET

Complex SWIFT messaging solution.

- Generic system for loading, storing and validating SWIFT messages
- The object model contains 323 SWIFT Message Types



SWIFT Framework .NET

SWIFT Framework .NET is a system for capturing, validating and processing SWIFT messages within an organization's information systems infrastructure. SWIFT Framework .NET allows seamless integration into existing IT systems and can be a base to build up a clean and fully compatible SWIFT messaging system.

SWIFT Framework .NET includes a generic system for the loading, storing and validation of SWIFT messages. It is also Microsoft BizTalk XML schema and object model compatible (without the need to install BizTalk itself). SWIFT Framework .NET enables validation based on message (or message block) properties and validation based on SWIFT tags.

These two validation methods can be combined and customized. Default message validation can be extended by your own specification or it can be completely replaced. This enables integrators to create simple business rules being built-in the messaging framework of the solution.

FROM SWIFT COMPONENTS

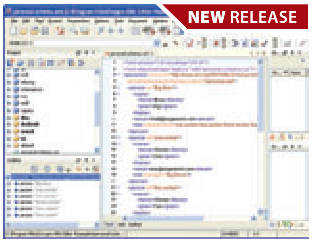
Product	Top 250	Review	Price
SWIFT Framework .NET	-	-	$4,410.00

SyncRO Soft

<oXygen/> XML Editor V12

Tools for XML authoring & conversion plus XPath, XSLT and XQuery development.

- Visual WYSIWYG XML editing mode based on W3C CSS stylesheets
- Grid editor allows you to edit repetitive XML content in a special layout similar to a spreadsheet application



NEW RELEASE

<oXygen/> XML Editor

<oXygen/> XML Editor provides the tools for XML authoring, XML conversion, XML Schema, DTD, Relax NG and Schematron development, XPath, XSLT, XQuery debugging, SOAP and WSDL testing.

Integration with the XML document repositories is made through WebDAV, Subversion and S/FTP protocols.

<oXygen/> XML editor supports the ability to browse, manage and query native XML and relational databases and supports a large number of text encodings including Unicode.

<oXygen/> XML editor is also available as an Eclipse IDE plugin, bringing unique XML development features to this widely used Java IDE.

FROM SYNCRO SOFT

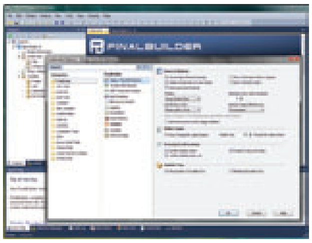
Product	Top 250	Review	Price
<oXygen/>XMLE ditor Enterprise V12	39	-	$440.02
<oXygen/>XMLE ditor Professional V12	149	-	$342.02
<oXygen/>XMLA uthor Professional V12	-	-	$195.02
<oXygen/>XMLA uthor Enterprise V12	-	-	$263.62
SyncroSV NC lient V5.1	-	-	$57.82

VSoft Technologies

FinalBuilder 7.0

Automated build and release management solution.

- Integrates with version control systems/Source code repositories
- Includes a broad range (600+) of action types that cover almost any task you might need to automate



FinalBuilder

FinalBuilder is an automated build and release management solution for Windows software developers and SCM (Software Change Management) professionals. Use FinalBuilder to define, debug, maintain, run and schedule reliable and repeatable build processes. FinalBuilder's user interface has been carefully crafted to make it as easy-to-use as possible. Anyone in your team can use FinalBuilder to automate tasks. You can compile applications from source code, compile setup installer programs, work with version control systems, handle versioning with ease, create and edit INI files and Windows registry keys, burn CDs and DVDs, or create CD/DVD images (ISO images), zip and unzip files and other archive formats, run automated testing, FTP files to/from servers, send emails, post on news servers, handle errors, run multiple tasks in parallel and dynamically change the build flow, iterate over sets of files and other lists and more.

FROM VSOFT TECHNOLOGIES

Product	Top 250	Review	Price
FinalBuilder 7.0	173	-	$351.82
Automise V3.0	-	-	$229.32
FinalBuilderS erver 7.0	-	-	$117.60

Charting for Visual C++ and ActiveX...

Advanced charting solutions for MFC, VB and .NET

Total Hours Worked
Hours
Employee
Worlds Most Popular Vehicle Colors
% of Color Sold
Vehicle Color
Top 10 States by Population
Population by U.S. State
Population Growth (Millions)
Decade / Year
Population by U.S. State
Decade / Year
Total Hours Worked
Hours
Employee

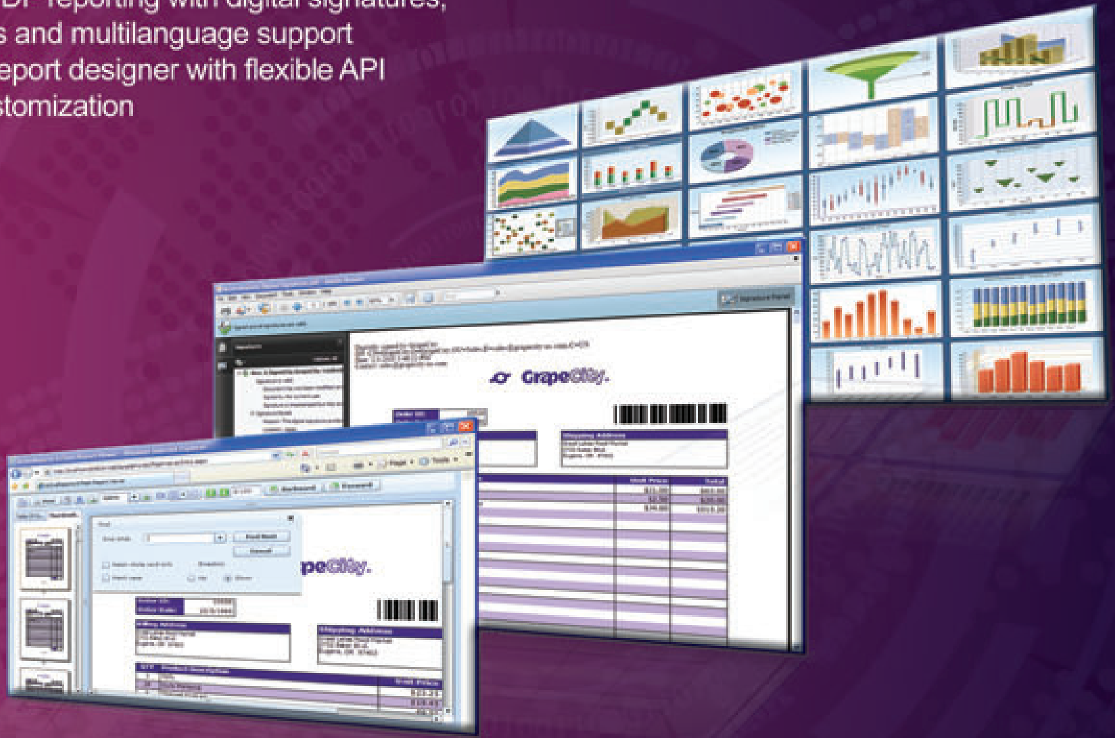
CODEJOCK
Chart Pro
for Visual C++ and ActiveX

Codejock's Chart Pro provides Windows developers with a sophisticated charting library that supports most of the popular chart styles used today. Many popular chart styles available including Area, Bar, Bubble, Candle Stick, Line, Fast Line, Funnel, Pyramid, Gantt, High Low, Pie, Point, Range Bar, Spline Area, Stacked Area, Stacked Bar, Stacked Spline, Scatter Line, Step Line, Stacked Spline Area, Doughnut, 3D Pie, 3D Doughnut, 3D Torus, 3D Pyramid and Rotated Bars.

Microsoft Visual Studio
PARTNER

Download a full evaluation from
www.componentsource.com

CODEJOCK®
software solutions


GrapeCity.