

# XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

January 2004 Volume: 5 Issue: 1

XML-JOURNAL.COM

## PREDICTIONS

**What's in Store for 2004**  
The benefits of XML will become even clearer  
pg. 3

## COMMENTARY

**Why We Need XML Query Standards**  
The perfect fit for a variety of XML environments  
by Jonathan Robie pg. 7

## STANDARDS

**Ontology and Integration**  
Managing application semantics using ontologies and W3C standards  
by David S. Linthicum pg. 46

pg. 25



REGISTER BY JANUARY 24, 2004

**SAVE UP TO \$300**

DISPLAY UNTIL March 31, 2004

\$6.99US \$7.99CAN



**SYS-CON MEDIA**

## Modernizing the Mainframe

Unleashing the power of XML and Web services 22

**Special in This Issue**  
IT luminaries share their insight on the future

## Policy – It's More Than Just Security

Toufic Boubez,

From JIT integration to Web services K. Scott Morrison & Maryann Hondo 8

## Enterprise Application Integration with a Native XML Database, Java, and Cocoon

Dan Hatfield

Berkeley DB XML brings powerful flexibility with a simple API 12

## Tree Structured Data and XML

Philip Burton & Russel Bruhn

Visualizing and transforming data with SVG and XSLT 18

## Modernizing the Mainframe

Joe Gentry

New technologies with legacy applications improve your bottom line 22

## Processing XML with C# and .NET

Andrew Solymosi

A solution that's simpler than you might expect 42

## Monitoring Air Pollution in Real Time Using XML

An open source answer

Rouslan Kadyrov & Kelly Carey 48



untangle your  
integration woes  
with

# Assande EAI Messaging Suite

## Assande EAI Messaging Suite features

- XML Schema Repository
- Data Abstraction, Transformation, and Mapping
- Message Assembly and Generation from Components
- Messaging Integration Service Deployment
- Automated Naming Standards, Version Control, and DIFF Functions
- Infrastructure Utilization and Pipeline Reporting
- Conversation Deployment and Message Release Management
- State Management and Integrated Workflow
- Advanced Search Capabilities
- Integration Documentation (MS Word, Excel) Repository

**ASSANDÉ™**  
INTEGRATION MANAGEMENT COMPANY

<http://www.assande.com>



## FOUNDING EDITOR

Ajit Sagar ajit@sys-con.com

## EDITORIAL ADVISORY BOARD

Graham Glass graham@themindelectric.com

Coco Jaenicke cjaenicke@attbi.com

Sean McGrath sean.mcgrath@propylon.com

Simeon Simeonov talktosim@polarisventures.com

## EDITORIAL

## Editor-in-Chief

Hitesh Seth hitesh@sys-con.com

## Managing Editor

Jennifer Van Winckel jennifer@sys-con.com

## Editor

Nancy Valentine nancy@sys-con.com

## Associate Editors

John Evdemon jevdemon@sys-con.com

Jamie Matusow jamie@sys-con.com

Gail Schultz gail@sys-con.com

Jean Cassidy jean@sys-con.com

## Assistant Editor

Kelly Flynn kelly@yachtchartersmagazine.com

## PRODUCTION

## Production Consultant

Jim Morgan jim@sys-con.com

## Art Director

Alex Botero alex@sys-con.com

## Associate Art Directors

Louis F. Cuffari louis@sys-con.com

Richard Silverberg richards@sys-con.com

## Assistant Art Director

Tami Beatty tami@sys-con.com

## CONTRIBUTORS TO THIS ISSUE

Stanko Blatnik, Toufic Boubez, Russel E. Bruhn,  
Philip J. Burton, Kelly Carey, Joe Gentry, Dan Hatfield,  
Maryann Hondo, Rouslan Kadyrov, David S. Linthicum,  
K. Scott Morrison, Jonathan Robie, Andrew Solymosi

## EDITORIAL OFFICES

## SYS-CON MEDIA

135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

TELEPHONE: 201 802-3000 FAX: 201 782-9637

XML-JOURNAL (ISSN# 1534-9780)

is published monthly (12 times a year)

by SYS-CON Publications, Inc.

Periodicals postage pending

Montvale, NJ 07645 and additional mailing offices.

POSTMASTER: Send address changes to:

XML-JOURNAL, SYS-CON Publications, Inc.,

135 Chestnut Ridge Road, Montvale, NJ 07645.

Worldwide Newsstand Distribution

Curtis Circulation Company, New Milford, NJ

## FOR LIST RENTAL INFORMATION:

Kevin Collopy: 845 731-2684,

kevin.collopy@edithroman.com

Frank Cipolla: 845 731-3832,

frank.cipolla@epostdirect.com

## ©COPYRIGHT

Copyright © 2004 by SYS-CON Publications, Inc. All rights reserved.

No part of this publication may be reproduced or transmitted  
in any form or by any means, electronic or mechanical,  
including photocopy or any information storage and retrieval  
system, without written permission. For promotional reprints,  
contact reprint coordinator. SYS-CON Publications, Inc.,  
reserves the right to revise, republish and authorize its readers  
to use the articles submitted for publication.

All brand and product names used on these pages  
are trade names, service marks, or trademarks of their respective  
companies. SYS-CON Publications, Inc., is not affiliated  
with the companies or products covered in *XML-Journal*.



## What's in Store for 2004

"What does the future hold for IT?" It's a burning question that everyone's asking, but who can answer? At *XML-Journal*, we went straight to the experts – the world's brightest and most prescient i-technology professionals – and asked them what they expect in the coming year for IT in general and XML in specific.



Industries stakeholders – software vendors, developers, technology investors, and users alike – are looking forward to the year ahead with renewed vigor. The buzz is around XSL, Web services, and the service-oriented architecture. The benefits of XML are clear...and in 2004 they'll become even clearer.

Following are some of the predictions we received from industry leaders...if they are an indication of what's to come, it's going to be a great year.

## Thom Robbins

Senior Technology Specialist  
Microsoft

2004 is going to be the year of the service-oriented architecture. We have talked and even debated over the design pattern for the last several years. This will be the year that it will finally become the standard design pattern of distributed scalable systems. I believe that we will see XML as the basis for this type of architecture as we design our applications. XML will finally permeate all aspects of server-side interchange using Web services to desktop-based XML. The benefits of XML are becoming clear and will become the basic design reality that we operate in when writing applications. This is the year that CIOs and other technology influencers will see their data no longer locked into proprietary formats and will use this to exchange data across disparate systems. The technology has finally come to the point where we can integrate these applications easily and clearly demonstrate a lower cost of ownership and quicker return on investment – the missing ingredients needed to finally take the industry into the next generation.

## Bill Rogers

CEO  
Ektron

We expect 2004 to be a landmark year for the adoption of XML content made available through Web services. More and more companies will design their Web strategies using this technology – whether it's in real estate, health care, finance, or government. XML and Web services will enhance the search, retrieval, exchange, storage, editing, publishing, management, use, and reuse of information.

In the coming year, Web services will begin to impact business results, and the way IT professionals and business users function in organizations, as the tool sets (.NET) for Web service technology are now mature. We believe the benefits of .NET will begin to be seen as IT professionals connect their companies more broadly and businesses streamline deployment time for creating applications. Additionally, the days of maintaining the same information in several applications will greatly be reduced, now that exchanging information becomes much easier.

## Bob Blakley

Chief Scientist, Security and Privacy  
IBM Tivoli Software

As pervasive computing and wireless networking penetrate the business IT market, partly as a consequence of an increase in telecommuting and remote working, the notion of a perimeter will become increasingly obsolete as a basis for IT and security architectures. Trusted end systems (both servers and, increasingly, clients) will supplant perimeter security solutions as this trend accelerates; this will drive demand for trusted-device standards such as TCG. The increased demand for these standards will result in their expedited convergence, the process for which has already begun.

## Tim Bray

CTO/Founder  
Antarctica Systems Inc.



I'm optimistic that we'll see more deals get done this year, but anyone who loosens their belt is nuts, because the economy is still very fragile and the nineties tolerance for excess and sloppy procurement culture is just totally over.

I think that interest in user-interface alternatives is really going to get legs in 2004; the state of the art in Web UI has been frozen for almost a decade now, and that just can't go on.

## Bill Weihl

CTO  
Akamai Technologies, Inc.



In 2004, XML Web services standardization will mature to a point that modular business applications will be consumed as on-demand services. Maturing standards for Web services plumbing and the advent of on-demand pricing models

will mean that this new generation of ISV-developed applications will be readily available for use in Internet applications deployed by enterprises. This will further expose the Internet's shortcomings as an e-business platform and drive the need for on-demand distributed computing solutions that deliver superior performance, security, and scalability.

## Raymond Schiavone

President and CEO  
Arbortext



In a survey we conducted this past summer, we learned that 69 percent of manufacturing companies expect significant to substantial improvements in reducing time to market as a result of XML-related technologies. Having taken numerous steps to streamline their manufacturing processes, this sector will now focus resources on eliminating inefficiencies to capitalize on cost and productivity improvements. Deploying XML-related technology to automate the way manufacturers capture, assemble, and publish information will be one of the primary steps this group takes - every-

thing from managing customer information to publishing maintenance manuals will be made more efficient with XML.

## David Kershaw

Professional Services Manager  
Altova, Inc.



Delivering XML-enabled applications requires teams that are equally expert in multiple arenas with high impedance mismatch: domain modeling, functional programming, relational data. Pulling all together is still tough even today.

2004 will be about making XML development more agile, creative, and productive by working out design strategies which we take for granted in the object-oriented programming world. XML will begin to leverage programming best practices to structure massive XML Schemas, create unit tests for XML and XSL, and clear up tangled concerns.

But at the same time, programmers will begin to realize that XML Schema brings stronger type and domain definition to the table and that XPath brings a common and high-powered rules language to any tree structure. At that point everybody starts to win.

***"I'm optimistic that we'll see more deals get done this year, but anyone who loosens their belt is nuts, because the economy is still very fragile"***

## Benjamin Chen

Chairman & CTO  
Snapbridge



It is clear that XML has become the de facto world standard for representing data and content. We believe that XML/XSL will supplant HTML within the coming years and that all content in the future will ultimately be presented from an

XML/XSL point of view. This means true reusability of data and content, hence notions of multichannel publishing, i.e., to print, Web sites, PDAs, cell phones, etc. In addition to revolutionizing the world of content, we see a massive wave of changes for the data side of the world. Enterprises have been dealing with the issues of getting to and utilizing silos of data grown about from the legacy of nonstandard means for representing data over the last three to four decades; with the advent of XML/XSL, we now have a standard lingua franca to truly begin integrating and sharing not only within, but between our enterprises.

The Internet has provided us with a universal highway to communicate across the globe. With the advent of XML/XSL we now have a common language with which to share both data and content. This will bring tremendous social and economic gains to our society, and at Snapbridge we believe that we are just beginning to see those changes taking effect.

## Ed Peters

President and Chief Executive  
DataDirect Technologies



The biggest story in 2004 will be ubiquity. The emergence of XML query language standards simplifies the exchange of data between relational databases and XML documents, and SQL/XML and XQuery eliminate the need for tedious

hand coding or for reliance on proprietary technologies that lock customers into one vendor's approach. With XML query language standards, it is much easier to share data across any platform, making data access a part of software development that can increase speed, performance, and the quality of the application.

## Dan Foody

CTO  
Actional Corporation



1. Web services security (WSS) standards will become standard functionality of any competitive Web services implementation.

2. When purchasing applications, 2004 will be the year when "Web services support" moves from nice-

to-have to must-have on customers' checklists. Vendors that lag in this support will be eliminated up front.

3. Despite hopes to the contrary, widely adopted standards for reliable delivery over Web services will not emerge until, at best, late in 2004.

## G. Ken Holman

CTO  
Crane Softwrights Ltd.



Now that RELAX-NG has become ISO International Standard 19757-2 in 2003, I'm anticipating a lot of vendors will be more comfortable embracing it and making it available to users as another alternative to modeling their XML information.

Being one part of the Document Schema Definition Language (DSDL) ISO 19757, it will mesh well with the many other parts in the standardization pipeline including an ISO version of Rick Jelliffe's Schematron and James Clark's Namespace Routing Language. When completed with its planned orchestration and flexibility features, the DSDL family promises to make a significant contribution to the community both as a whole and by its individual parts.

New XSL releases of XPath 2, XSLT 2, and XSL-FO 1.1 will address many areas where workarounds have been used in stylesheets for years. Convenience features and new functionality will make many stylesheets slimmer and faster. New and powerful composition features in XSL-FO will enhance the already professional looking results available to users who need to paginate and print their XML information. As evidenced by the increased interest we are seeing in XSL training around the world, more and more people are exploring and adopting stylesheet technologies for the transformation and presentation of their structured data.

### Eric Newcomer

Chief Technology Officer  
IONA



I think in 2004 we will begin to see the emergence of XML as a GUI replacement technology. I think we will also start to hear some rumblings about XML as a replacement for SQL as well. Just on the cost basis alone, developing GUIs and designing, developing, and administering databases represent anywhere from one-half to three-quarters of total project cost. I expect the continuing price pressure on IT to expand the interest in additional applications of XML such as using it in place of a GUI or database. Documents are a more natural way of interacting with a computer than structured data, and with the widespread adoption of Web services for interoperability between devices and servers, I expect XML to be driven also toward more applications on the devices and servers themselves.

### David Litwack

Senior Vice President  
Novell



Web services are increasingly being used to repurpose existing systems by creating composite, logical applications in front of physical systems. These may sometimes be SOAP/WSDL based and almost always will be XML.

The result is a logical data source constructed from one or more physical systems that represent a business function as perceived by some audience. This presents to the user an application that looks like it was developed just for them – everything they need to do their job and no more than they need.

### Jeff Mischkinsky

Director of Web Services Standards  
Oracle

On the standards front, 2003 was the year we as an industry finally began cutting through the confused clutter of Web services specifications and started to deliver on the promise of truly open and interoperable Web services. In August, the WS-I, co-founded by Oracle, published the Basic Profile 1.0, giving companies and developers a common set of guidelines and conventions for implementing interoperable Web services. With the basic profile successfully demonstrating that a large and diverse group of companies could work collaboratively together and compromise in the common interest, a number of vendors jointly began building industry consensus around developing open royalty-free standards for the rest of the Web services stack. This helped to ensure that adoption of Web services will not be hindered by concerns over proprietary technologies or submarine patents.

Following Oracle's successful efforts to revise the Java Community Process participation rules to make them more uniform and accessible and our work with the W3C's patent policy to clarify the royalty-free IPR status of the W3C's Web services specifications (such as the recently released SOAP 1.2 recommendation), we worked with other leading Web services vendors to charter a number of OASIS technical committees (TCs) based on royalty-free specifications and

processes. The Web Services Reliable Messaging TC has already begun hosting proof-of-concept interoperability demonstrations, most recently at XML 2003 in Philadelphia. The Web Services Composite Application Framework TC was chartered to provide an open framework for managing context and coordinating transactions across multiple Web services; expect to see early implementations and proof-of-concept demonstrations next year.

### Bill Ruh

Senior Vice President of Professional Services  
Software AG, Inc.



I believe that 2004 will be the year that XML begins to make headway among mainstream users. In 2003 we saw a growing number of tools for creating schemas, managing schemas, and taking proprietary metadata out of databases and turning it into XML. These tools are important to the infrastructure practitioners.

Now we are seeing the next phase, whereby the tools being used every day in the typical office environment are starting to migrate to XML. This process will accelerate acceptance of XML among a broader audience. For example, Microsoft's newest version of Office is XML enabled and includes a forms capability. This allows a user to create a form that is XML enabled and publish that form to a central location where others, let's say co-workers in the same department, can fill out and submit the form. This new type of capability is going to empower the user.

People will discover, for example, that using XML they can pull data out of their databases and put it right into Excel. And those who don't have that infrastructure will begin calling for it, "I've seen that. Somebody told me I could do that." So we will see the demand for XML-based infrastructure start to come from the ground up – from individual users. That's what we saw in the past with the Ethernet TCP environments; that's what we began to see with the Web environments. The same thing is going to happen with XML, beginning in 2004 and gaining momentum during the next several years.

### Sinisa Zimek

Director  
NetWeaver, Standards Strategy  
SAP Labs, LLC.



XML and Web services will further gain momentum in 2004. A whole variety of new industry standards will emerge; standards for ensuring security, maintaining higher reliability and such for allowing easier and better orchestration of Web services. At the same time, existing standards like SAML, WS-Security, and BPEL will find broader adoption and maturity. Standard bodies like W3C and OASIS will remain in the spotlight in 2004; the role of standards integrators like WS-I will increase significantly due to best practice and guidance services they provide to vendors and implementers of Web services. On the application side, the number and functionality of Web services applications will increase significantly during the next year.

***"As Web services move from inside the corporate firewall to higher-value, mission-critical Web services, expect to see Web services management solutions that are fully SOAP aware from the major management vendors"***

### Doron Sherman

CTO  
Collaxa, Inc.



Year 2004 will see a dramatic increase in the use of the BPEL (Web service orchestration) standard for automating workflow and business process management applications. Sun will gather wide support for its JBI (Java Business Integration, aka JSR 208) architecture, the Java answer to address the fragmented and proprietary business integration market. The push for JBI will further accelerate adoption of the BPEL standard by enterprise users. Deployments of Web service applications, including BPEL, will also increase use of WS-Security as an integral part of the application infrastructure.

### Dan Ryan

Executive Vice President of Marketing and Business Development  
Stellent

We expect to see more organizations use Web services for integrating content and content management functionality into other enterprise applications, whether they are packaged applications or developed internally. Similarly, we are seeing a trend whereby partners develop value-added applications that integrate content management through the use of Web services. The Stellent product suite has always been built on a services-based architecture, and Web services are an elegant and cost-effective means for accessing and integrating our functionality.

### Paul Lipton

Technology Strategist, Office of the CTO  
Computer Associates



Web services management will make major strides in 2004 with the completion of the OASIS WSDM (Web Services Distributed Management) specification and a broader understanding in IT shops that Web services truly need management, especially in a B2B setting. As Web services move from inside the corporate firewall to higher value mission-critical Web services, expect to see Web services management solutions that are fully SOAP aware from the major management vendors like CA. ☉



Introducing

# SOAPscope 3.0

# Debug Test Tune

## 4 Ways to Know Your Web Services

Whether you are learning how a Web service works, or troubleshooting a tough problem, you need the help of a "smart" tool. SOAPscope lets you dig deeper, faster.

**Try It** Solve problems by testing your Web service with different inputs without writing any code.

**See It** View WSDL and SOAP to understand what's happening. Capture from any toolkit, and see just the right detail for the task at hand.

**Diff It** Compare a problem message or WSDL with a similar, working one.

**Check It** When the problem's not obvious, rigorous interactive analysis finds inconsistencies, errors, and interoperability problems.

## Look What's New in 3.0

- Microsoft® Visual Studio® .NET Integration
- Graph Message Statistics
- Interactive Message Analysis
- Interoperability Testing System
- SSL Support
- HTTP Authentication Support
- HTTP Compression Support
- Support for multi-byte encoding
- Best usability of any Web Services tool

The most comprehensive Web services diagnostic system available, and still **only \$99!**

*"Comparing SOAPscope and other SOAP sniffing tools is like the difference between shooting a bullet and throwing it."*

Scott Hanselman  
Chief Architect  
Corillian Corp.

**Mr. SOAPscope says -  
"Try SOAPscope free at  
[www.mindreef.com](http://www.mindreef.com)!"**

**Try It** online at [XMETHODS.net](http://XMETHODS.net) or download a trial at [Mindreef.com](http://Mindreef.com)

© Copyright 2004, Mindreef, Inc. The names of companies and products mentioned herein may be the trademarks of their respective owners. This product uses Hypersonic SQL. This product includes software developed by the Politecnico di Torino and its contributors.



"The Web Services Diagnostic Experts" **Mindreef**

**PRESIDENT and CEO**

Fuat Kircaali fuat@sys-con.com

**VP, Business Development**

Grisha Davida grisha@sys-con.com

**Group Publisher**

Jeremy Geelan jeremy@sys-con.com

**Technical Director**

Alan Williamson alan@sys-con.com

**ADVERTISING**

**Senior VP, Sales & Marketing**

Carmen Gonzalez carmen@sys-con.com

**VP, Sales & Marketing**

Miles Silverman miles@sys-con.com

**Advertising Director**

Robyn Forma robyn@sys-con.com

**Director of Sales & Marketing**

Megan Mussa megan@sys-con.com

**Advertising Sales Manager**

Alisa Catalano alisa@sys-con.com

**Associate Sales Managers**

Carrie Gebert carrieg@sys-con.com

Kristin Kuhnle kristin@sys-con.com

Beth Jones beth@sys-con.com

**SYS-CON EVENTS**

**President**

Grisha Davida grisha@sys-con.com

**Conference Manager**

Michael Lynch mike@sys-con.com

**National Sales Manager**

Sean Raman raman@sys-con.com

**CUSTOMER RELATIONS**

**Circulation Service Coordinators**

Shelia Dickerson shelia@sys-con.com

Edna Earle Russell edna@sys-con.com

**JDJ STORE**

**Manager**

Brunilda Staropoli bruni@sys-con.com

**WEB SERVICES**

**VP, Information Systems**

Robert Diamond robert@sys-con.com

**Web Designers**

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

**Online Editor**

Lin Goetz lin@sys-con.com

**ACCOUNTING**

**Accounts Receivable**

Charlotte Lopez charlotte@sys-con.com

**Financial Analyst**

Joan LaRose joan@sys-con.com

**Accounts Payable**

Betty White betty@sys-con.com

**SUBSCRIPTIONS**

**SUBSCRIBE@SYS-CON.COM**

1 888 303-5282

For subscriptions and requests for bulk orders,  
please send your letters to Subscription Department

Cover Price: \$6.99/issue

Domestic: \$69.99/yr (12 issues)

Canada/Mexico: \$89.99/yr

all other countries \$99.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

# Why We Need XML Query Standards

WRITTEN BY JONATHAN ROBIE



Despite a shortage of sophisticated XML query tools, Internet demands have forced companies to present their data in various formats. In one sense little has changed, as SQL queries have long been used to combine data for different purposes and audiences. Now however, the output is XML, and while the tools used to generate or consume XML may be different, the structure is similar to structures created by traditional report writers.

Relational databases have neither hierarchy nor sequence, but XML uses both to structure data. This is a good match for the way XML data is used. Most Web sites use data from relational databases, though it rarely looks relational on a Web page. Nobody wants to present users with a series of two-dimensional tables, telling them how to join the tables in their minds to see the relationships. Instead, relational data is used to build hierarchical representations of the data for users. Similarly, most Web messages have a strongly hierarchical organization that looks nothing like the relational tables. For both Web sites and Web messages, data is generally exchanged as XML. Web sites convert the XML to HTML using stylesheets, and Web messages exchange data directly as XML.

Companies basically have four choices when deciding how to use relational data in XML applications - custom coding, proprietary XML extensions from database vendors, SQL 2003's SQL/XML extensions, and XQuery.

Any of these approaches can be used, but they have very different ramifications for the architecture of software systems.

Custom programming is probably the most widespread approach to integrating XML and relational data. These programs use JDBC or ODBC to issue SQL queries to query the database, create XML structures using APIs like DOM or SAX, and use XSLT to transform structures or format XML for display as HTML.

This approach is standards-based and portable but requires much tedious coding, and the same information is often represented in several intermediate formats. This code is difficult to maintain when new formats are required or existing formats change. It is tricky to make these applications perform well, and not usually cost effective since new programming is needed for each desired format.

Relational vendors have long recognized the need for integration with XML, providing SQL

extensions and other tools for their products. These tools vary widely in quality, performance, and usability and have been important for the evolution of our understanding of XML query languages. For companies that can afford to rely on only one database vendor, proprietary tools can be a good alternative to custom coding, though they are inherently nonportable.

Recognizing the need for a standard for adding XML support to relational databases, INCITS, ANSI, and ISO have added XML publishing functions to SQL 2003. These functions are easy to learn and allow any desired XML structure to be created. As part of the SQL language, the full power of SQL is available to structure data; for an experienced SQL programmer, this is the simplest tool that solves the problem well, and it fits well into existing relational infrastructure. Portable implementations of SQL/XML are commercially available, using the standard JDBC API to access query results.

The W3C has designed the XQuery language to query XML in the same way that SQL queries relational data. Input and output of an XQuery are XML, and XQuery works efficiently for XML views of data sources that are not represented as XML. Most relational vendors are implementing XQuery based on SQL/XML views of their relational tables, and some third-party tools can provide XML views of any relational database, allowing data to be combined with XML files or XML represented using SAX or DOM in a program. Likely to become a Recommendation soon, XQuery easily combines multiple data sources making it a clear winner for data integration.

...

Both XQuery and SQL/XML are extremely useful for businesses that need a portable way to query their data to produce XML. XQuery is more powerful for data integration, and fits better into many XML environments. SQL/XML fits well into existing relational infrastructure, and requires little new learning from programmers.

## AUTHOR BIO

Jonathan Robie is the XML program manager at DataDirect Technologies. Before joining DataDirect, Jonathan was an XML research specialist at Software AG. Jonathan works very closely with the W3C; he is a co-author of the XQuery specification, has participated in several W3C Working Groups, and speaks regularly at XML conferences. Jonathan wrote an XQuery tutorial for a book called *XQuery from the Experts* which is now available on Amazon.com.

JONATHAN.ROBIE@DATADIRECT-TECHNOLOGIES.COM



HOME



ENTERPRISE SOLUTIONS



CONTENT MANAGEMENT

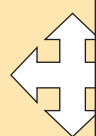


DATA MANAGEMENT



XML LABS





WRITTEN BY

TOUFIC BOUBEZ,  
K. SCOTT MORRISON  
& MARYANN HONDO*From just-in-time integration to Web services*

**B**usiness has long pursued the goal of making IT more of a strategic tool and less of a necessary evil. Organizations are constantly looking for easier, cheaper, and more logical ways to build applications and unite the silos of functionality they still depend on. One approach that has met with some success is the concept of just-in-time integration – a technique to combine new functionalities as quickly and cheaply as required, whether they reside inside an organization or outside of it (i.e., with a business partner).

From the architectural perspective, just-in-time integration is a cornerstone of service-oriented architecture (SOA). Under SOA, applications consist of aggregations of calls to services. Services are simply coarsely grained functions that are made available to invoking applications using a consistent semantic. They might encapsulate a well-defined unit of business logic, a legacy application, or an interface to a data gathering system. What a service does is not a concern of SOA – how it is made available is. One of the fundamental principles of SOA is the idea of loose coupling. Loosely coupled systems exhibit a flexibility such that a change effected in one component of the system does not break the rest of the system.

To achieve this, SOAs typically provide a mechanism to publish services and a means for consumers to discover and invoke them dynamically. Web services is an SOA – composed of technologies like SOAP, WSDL, and UDDI – that has the unique characteristic of being based on open standards and being independent of the deployment platform. This is in contrast to other SOAs, such as Sun's Jini, and alternative distributed application technologies, such as OMG's CORBA or Microsoft's COM+. But despite the media triumph of Web services, we still have a long way to go from this basic set of technologies to the end goal of just-in-time integration. This article will explore one of the most important issues in providing true loose coupling in a system of interconnected services.

**WSDL Isn't Enough**

A fundamental element in SOA is the interface, or contract. It defines the syntax of a service. It describes an interface by name, what data types the consumer must provide when call-

ing the service, and what a consumer can expect to receive in return. The contract may also describe some service semantics using comments embedded in the description or through the logical grouping of functionality – such as methods or operations – into a common service unit. There is congruency between the SOA contract and the interface defined in languages like Java or C#, though unlike Java no semantic clues can be derived from static final variables, which are not exposed as part of a service definition.

The contract in Web services is the WSDL document. WSDL is a powerful technology, but it's important to recognize its limitations. WSDL describes an interface in two ways. It provides an abstract description of types and messages, but it also includes at least one concrete description that binds the abstract definition to a particular messaging technology. At present, this is almost exclusively HTTP SOAP messaging; however, WSDL, being XML, is reasonably extensible and so could similarly declare concrete bindings to other transport and marshalling mechanisms, such as proprietary binary protocols.

Since WSDL's limitations are not immediately obvious, let's consider a simple scenario familiar to everyone. Suppose you are a developer charged with building and deploying the corporate getQuote service (sorry). Disregarding, for a moment, the other important characteristics of SOA you might prefer to explore, focus instead on the everyday deployment: simple, point-to-point Web services. It would seem on first examination that WSDL provides everything we need to tie our client and quote server together. We have the service and operation names, message parts, data types, a return value – largely the same as we get with a Java or C# interface. Which brings us to an important question: Is WSDL no more than syntax – the morphology of the message needed to invoke the service remotely in an SOA? Is this all we need to achieve loose coupling in our architecture?

In some circumstances, the answer is clearly yes: we've all deployed and invoked getQuote with no more than this at our disposal. However, in doing so we may fall into a common trap, relying on WSDL for more than it was intended for. Suppose your next task is to deploy a secure getQuote using SSL.



This is easy enough to accommodate in WSDL: a simple modification to the URL signifying the transport is https may be all that is necessary (in WSDL, the URL appears in the address element as the value of the location attribute). Unfortunately, your secure getQuote is destined to become a victim of its own success: over time, it increases in popularity to such an extent that it captures the notice of company officials. Shortly thereafter, a new corporate fiat comes down from the chief security officer (CSO) declaring all secure external communication is to be encrypted using 168-bit 3DES-EDE-CBC (Triple DES Encrypt-Decrypt-Encrypt Cipher Block Chaining – not a fast block encryption algorithm, but a very secure one).

SSL can accommodate this; however, WSDL cannot. This means you have to open up your code and start making changes. If you're lucky you might be able to simply tweak your infrastructure settings and let SSL negotiation run its natural course. If not, you might have to explore deep within your SOAP development kit and substitute an entirely new encryption layer (good luck...). Needless to say, these changes will break any client applications that rely exclusively on the original WSDL document for their interface and now have no way of discovering the updated security requirements. Our once loosely coupled system has become quite tightly and mysteriously bound.

However, we still aren't finished with getQuote. A further complication arises when this service is made available to a diverse group of service requesters, but subject to different security and reliability profiles. Those applications residing in the internal security domain must authenticate with the server using client-side certificates, which have been deployed recently as part of the corporate PKI initiative. IT is lagging in deploying certificates on external systems, so these systems should authenticate with username and password under the HTTP digest mechanism.

Herein lies a host of problems for the Web service provider. From the specification perspective, how do you convey these alternative requirements? From the development point of view, how do you implement these options into the service? From the maintenance point of view, how do you make subsequent changes in the service as new or different (but equally amorphous) requirements materialize, all without impacting existing service dependencies?

The service requester also faces considerable burdens: a vague yet complex set of security requirements defined by the provider, a set that is well beyond the capabilities, or indeed the intent, of WSDL to capture. Certainly, it eliminates any possibility of using a WSDL stub (or proxy) generator, so popular in modern IDEs. It is a situation that conspires to make it difficult to develop loosely coupled systems.

### Formalized Policy Documents in SOAs

While this sounds like a contrived example, it actually illustrates a problem that's more common than you might think. Working within the context of a single process space, there is much we get free, so naturally there is much we take for granted. Security context (the state that declares who we are and what resources we can access), for example, is maintained by the OS and calls to methods don't cross process boundaries so there are no integrity or confidentiality issues to be considered. Linkers ensure that the code for a module is loaded and executed correctly and transparently. Distributed applications, in contrast, come with few of these luxuries and a whole host of new security risks that demand to be addressed.

The danger with simple Web services is that they lull us into overlooking the fundamental differences between a local and distributed application. In a way, they are too easy. Web services are brilliantly simple to deploy, and in delegating to exist-

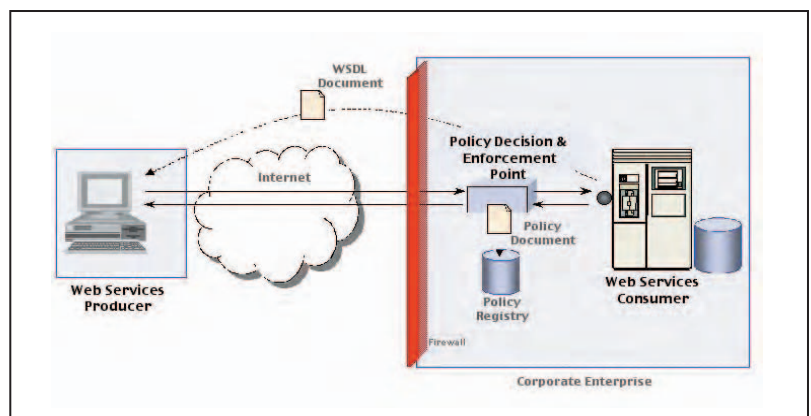


Figure 1 • Policy engine

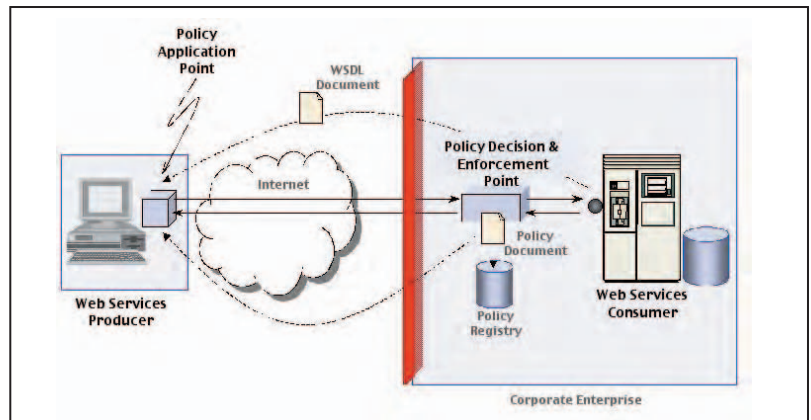


Figure 2 • Agent application

ing and familiar infrastructure possess a resonance of completeness. But these solutions gloss over a more deep-seated problem: what we actually need is a mechanism to control how clients talk to services that is more rigorous, more expressive, than an IDL such as WSDL. What is required is finer control over the entire variant parameter space of the conversation – a difficult task when this is concealed in the murk of largely invariant code. This challenge begins with security-related parameters; however, it extends to a number of other equally critical parameters common to most robust distributed applications.

Consider the complexity of defining something as fundamental as the basic security model for a SOAP transaction. We could choose to delegate security to channel-based technologies (such as SSL or a VPN), or follow the newer trend espoused by the WS-Security initiative and decouple basic security from transport, only to absorb it in the message itself by signing and encrypting relevant message parts. How should our application perform authentication: present basic credentials; respond to a digest challenge; or use its client-side certificate? Should a consumer authenticate each message individually or maintain a security context, tracking this with session IDs or the continuity inherited from a persistent socket?

Historically, such questions have very often been answered implicitly through coding – a technique that rarely acknowledges or articulates its dependencies and underlying approach. Woe betide the poor developer charged with making a policy change to the security model hard coded deeply within the fabric of an application. More often than not, the fabric unravels.

What we would like in Web services – and by extension, SOA in general – is a way of declaring such policy directives

independent from implementing the actual application code. Furthermore, the statements we make about policy should be dynamically bound to an application at runtime – a concept entirely consistent with the SOA directive of late binding. Need to assert a longer key length for encryption? Make the change to the policy declaration and it should be bound instantly to the running application. Policy, therefore, delivers on the promise of true loose coupling.

Clearly, WSDL today, with its focus on the syntax of the interface, isn't up to this. Fortunately, such an effort has been started, and a first proposal has been published as a framework of WS-Policy specifications.

## Beyond Security

Policy suffers from being an overloaded word. Within the discipline of computer security, it has a particular multiplicity of meaning. To a CSO for example, policy is a set of corporate-specific security principles documented in a binder. This might follow the formal guidelines laid out in Internet RFCs 1244 and 2196, which describe how to develop a comprehensive security policy for an organization. As roles in an organization become more finely grained, so too does the scope and interpretation of policy. The firewall administrator laboring in the CSO's business unit probably thinks of policies in terms of concrete rules applied to TCP and UDP ports. The human resources department, on the other hand, defines policy regulating sick leave, carrying over vacation, which employees get a parking decal. Policy here often has nothing whatsoever to do with security issues.

With distributed computing, we very often see policy only in terms of security, missing its other nuances. Reliability is a good example of a characteristic of transmission that can be managed declaratively through policy. Large corporations like financial institutions spend vast sums of money on message-oriented middleware (MOM) to ensure that messages – and some of these are SOAP messages – are guaranteed to be delivered from one system to another, even if the ultimate destination is down when the message enters the system. Anyone who has built applications for these systems appreciates that MOM offers a tremendously powerful programming paradigm. It's inherently asynchronous, loosely coupled, and extremely scalable.

One of the beauties of MOM is how little a developer needs to know about the actual configuration of the infrastructure. Message time to live, routings through the network, retransmission periods – all of these parameters are configured by an administrator, as opposed to being hard coded into the application. An administrator can make radical changes to server deployment or alter the path a message follows in a WAN with no modification to client or server code. No developer needs to be engaged, and testing time can be greatly reduced.

MOM demonstrates the power of effective parameterization of distributed computing. Web services policy can bring similar profits. But rather than simply parameterizing the local operating characteristics of an application, as we so often do with property files or OS registries, it's about parameterizing all the characteristics of Web services transactions. Security clearly is the beginning, but reliability, transactional characteristics, Quality of Service (QoS) guarantees and expectations, and even message transformational parameters are all equally valid to decouple from an application and to make administrable.

## Service Provider and Consumer Support

To support the SOA concept of runtime binding, policy

must seamlessly integrate with existing Web services infrastructure, and be applicable across existing transactions. This is where the concept of a policy engine becomes attractive, as shown in Figure 1. A policy engine is a critical piece of Web services infrastructure that provides a point to administer, register, and apply policy to incoming Web services transactions.

Simple enforcement of policy is sufficient for some very basic security parameters that rely on unambiguous application of standards. For example, if policy only requires that transactions provide HTTP basic authentication credentials, it is something that's reasonable to delegate to the developer of the client. This is a clear requirement, easily written down and widely supported in existing SOAP development kits. In this role, the policy engine is little more than a simple application-level firewall.

However, the moment policy becomes more sophisticated and more dynamic, we have a problem. How do we effectively communicate complex policy requirements, which may continuously shift in response to changing business needs, to the client developer? How does the developer, using the Apache SOAP toolkit, accommodate a requirement for reliable message transmission, XML signature and canonicalization of the message envelope, and XML encryption of the body element? What happens when the OASIS WS-Security specification changes slightly, rendering our message traffic noncompliant?

Clearly, as policy grows in sophistication, the need for a Policy Application Point becomes critical. This can manifest as an agent application, shown in Figure 2, that can interpret policy documents, and dynamically apply policy assertions to Web services transactions at runtime. As policy changes, so too does decoration of a message in accordance with that policy. The agent can transform a message to be in compliance with the latest security specification; it also provides the mechanism for reliable messaging. This is the true promise of declarative policy. It realizes the dream of loose coupling by removing platform dependencies and assumptions from application code and putting the enforcement of policies into the hands of administrators, who can customize these in direct alignment with current business needs. ☛

### AUTHOR BIO

*Toufic Boubez is a well-respected and renowned Web services visionary. Prior to cofounding Layer 7 Technologies, Toufic was the chief Web services architect for IBM's Software Group and drove their early XML and Web services strategies. He coauthored the original UDDI API specification, and his most recent book is the top-selling Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI.*

*K. Scott Morrison provides technical leadership to the SecureSpan Solution development team. Prior to joining Layer 7 Technologies, Scott was the director of architecture and technology for Infowave Software. Scott is a frequent speaker at technology conferences and has been published extensively in leading academic and computer trade journals. He is also the co-author of Sam's Java Web Services Unleashed.*

*Maryann Hondo is the security architect for emerging technology at IBM, concentrating on XML security. She is one of the coauthors of the WS-Security, Policy, Trust and Secure Conversation specifications announced by IBM and other business partners. Before joining the emerging technology group she managed the IBM Tivoli Jonah team (IETF PKIX reference implementation) and was security architect for Lotus e-Suite participating in the development of Java Security (JAS).*

TBOUBEZ@LAYER7TECH.COM

SMORRISON@LAYER7TECH.COM

MHONDO@US.IBM





Everything **OLD** Is **NEW** Again!

Imagine **XML-enabling** your existing **PowerBuilder** and **Visual Basic** applications

**QUICK** In hours, without changing a line of code  
**CHEAP** Inexpensively, using your current skills  
**SAFE** Safely, preserving all logic and security

**NOW YOU CAN**

[www.active-endpoints.com](http://www.active-endpoints.com)



# Enterprise Application Integration with a Native XML Database, Java, and Cocoon

## Powerful flexibility with a simple API

A client recently asked EDS to design and support an EAI implementation based on XML messaging. The implementation of this solution created a need for an internal application that would allow multiple developers and analysts to create and manage a variety of XML documents. The solution needed to be inexpensive, flexible, extensible, and yet easy to manage.

Since the client's environment and business needs are constantly changing, the company didn't want to spend excessive time on support and maintenance of internal applications. The solution also needed to leverage the effort EDS put into creating and maintaining XML documentation around these data services.

To create the solution, EDS used two open-source technologies – Sleepycat's Berkeley DB XML (bdbxml) ([www.sleepycat.com](http://www.sleepycat.com)) and Cocoon (<http://cocoon.apache.org>). The Apache Cocoon product provides the XML-based application framework, while the bdbxml product is an application-specific data manager providing XML persistence and retrieval. The combined technology creates a flexible, powerful framework for quickly building and deploying customized XML-based applications.

### The Software

Apache Cocoon, an XML publishing framework, employs a dynamic pipeline-processing concept that generates, manipulates, and ultimately renders the XML content into the presentation format of the user's choosing. Generic templates exist for rendering XML to standard formats such as HTML and PDF. The framework is easily configurable and extensible for custom XML-

based applications. A significant portion of this configuration and customization is accomplished through XML documents that detail the pipeline steps necessary for a custom application.

The Cocoon pipeline consists of three types of components – generators, transformers, and serializers. A generator, the starting point for any given pipeline, produces the XML that is processed by the pipeline. Transformers can manipulate the XML as needed or perform operations based upon the XML content. Multiple transformers can be employed as needed. Finally, a serializer renders the final XML content (the result of the last transform) into a presentation format. The XML is passed through the pipeline as SAX events.

From a content application perspective, EDS used Cocoon to deliver dynamic content to analysts who were researching or testing our XML services. Conceptually, this process is common, and several products would have sufficed.

The Cocoon project provided the additional benefits of being:

- Flexible and extensible
- Open source
- Free

With an XML-driven application server like Cocoon, we also found we needed an XML storage medium. Apache has built-in support for the Xindice XML product. However, Xindice is a pure-Java XML database with a Java API. We found we wanted support for non-Java APIs. Additionally, we desired an application-specific data manager, which allows flexible integration at various architectural levels.

This criterion led us to Sleepycat's bdbxml.

### Introducing Berkeley DB XML

Sleepycat's Berkeley DB XML product (bdbxml) is built on the company's Berkeley DB product. It is designed to store and retrieve XML documents and provides XPath querying capabilities against the set of stored documents. The multiplatform product provides APIs in a variety of languages. It also can manage both Berkeley DB and bdbxml data under the same "container." For the EDS solution, our team focused on the bdbxml Java API and XML capabilities. The product leverages the open source Pathan binaries for XPath functionality and the Xerces-C++ binaries for XML parsing capabilities. These libraries provide native-platform speed and are integrated well into the product.

The bdbxml product is open source as well, under a dual-style license. Customers can use it at no charge if the embedding application is open source or used only at a single physical site. Customers that redistribute bdbxml within a non-open source product must purchase a commercial software license. The API is simple to use, consisting of seven base classes, and a great deal can be accomplished without understanding much more than this simple XML interface. If more robust techniques are necessary then the full API provides access to objects that manage the complexity in indexing, transactionality, concurrency, and distributed transactions as necessary.

For EDS, the product met the necessary requirements by providing:

- Flexibility by storing the native XML documents and providing XPath query capabilities against them
- Round-tripping capability to a simple repository
- Simple document-oriented metadata

### AUTHOR BIO

Dan Hatfield is a senior consultant within EDS' EAI practice. He uses XML and Java technologies extensively in this role. He often leverages open source solutions to improve project infrastructure and team productivity.



that can be stored without affecting the integrity of the XML document

- Multiplatform and multilanguage APIs
- A lightweight implementation that is highly efficient for XML document storage and retrieval
- An application-specific nature providing flexibility in future architectures and application

## The Solution's Data Model

The solution uses bdbxml as the XML source for the application framework. The EDS custom-built interface to bdbxml is through an XML document that provides instructions to the component, which encapsulates the interaction with the database. This architecture is extremely flexible within the Cocoon framework and allows for substantial reuse of this component in various pipelines. The downside to this decision is that there is overhead in interacting with the component via an XML document; however, bdbxml's superior performance more than makes up for this shortcoming.

The application's traffic will not be high volume, and the solution that it employs provides a separate "database" for each user. While this would not be considered with a traditional database, the application-specific nature of bdbxml makes this feasible. This architecture simplifies the solution enormously. By creating one database for each user, indexing and performance concerns are eliminated, because the container will never grow that large for a single user (based upon the requirements).

Transactionality and ACID concerns are removed because users will be limited to updates against their own database. A Cocoon application can store and retrieve these documents without having to worry about the complexities around synchronizing access to a common resource.

## Getting Started

To create a similar solution, you should have JDK 1.4 to work with as well as Cocoon (I used Java.1.4.0\_02).

The first steps are downloading and installing Cocoon and bdbxml. You may want Ant installed (<http://ant.apache.org>) if you want to use the Ant scripts provided.

To download and install Cocoon, visit [cocoon.apache.org](http://cocoon.apache.org) and follow the instructions. Choose version 2.1 or later. The download and installation process is straightforward. You may select either the binaries or the source version of

Cocoon. If you choose the source version, you will need Ant installed to build the binaries.

To download and install bdbxml, visit [www.sleepycat.com](http://www.sleepycat.com) and select the Berkeley DB XML product. The installation of bdbxml is simple. You will need to compile the source. If you run into problems, go to [www.merrells.com/john/dbxml](http://www.merrells.com/john/dbxml).

It would also be helpful to obtain the Getting Started Guide available at [www.sleepycat.com/products/pdfs/GettingStartedJava.pdf](http://www.sleepycat.com/products/pdfs/GettingStartedJava.pdf). Once the products are installed and compiled, you're ready to begin.

## A Brief Look at the Java API

Before working with the custom Transformer code, take a quick walk through the bdbxml API. Javadoc is included with the installation, and it may be helpful to browse this too.

The XML Java API resides in the `com.sleepycat.dbxml` package. The `XmlContainer` object encapsulates the concept of a database in bdbxml. It provides methods for opening, closing, loading, and saving (the `dump` method) the database, as well as creating indexes. It also provides methods for extracting documents via XPath (`queryWithXPath`) and the document ID (`getDocument`). Documents are added to the container via the `putDocument` method and removed using the `deleteDocument` method. No update document method is provided. Updates are accomplished by first deleting and then re-adding the document. The `XmlDocument` object encapsulates an XML Document and related metadata.

This class provides methods for retrieving and storing the content of the document as a string and as a byte array. It also provides methods for getting and setting metadata describing the document. This nifty feature allows users to store metadata about the document without affecting the integrity of the XML document.

The `XPathExpression` class is used to provide an object representing a parsed XPath expression. These objects can be obtained from the `XmlContainer.parseXPathExpression`. They are passed to the `XmlContainer.queryWithXPath` method to query the database for documents matching the XPath expression.

The `XmlResults` and `XmlValue` objects provide a simple interface for examining results of the XPath query against the database. The `XmlResults` object is returned from the `XmlContainer.query-`

`WithXPath`. It is similar in fashion to an iterator with a `size` method to obtain the number of results and a `next` method to retrieve the "next" result. A call to this `next` method returns an `XmlValue` object.

The `XmlValue` object encapsulates a node in an XML document. XML document nodes can have one of three value types – boolean, string, or number. `XmlQueryContext` is a configuration class used to control the behavior of the `XmlContainer` when executing XPath queries. The query mechanism supports both eager and lazy querying, as well the capability to return either full documents or document fragments.

The `XmlIndexDeclaration` class is used to define an index declaration. Indexes can be created by methods on the `XmlContainer` object. The type of index to create is situation specific.

## XML Interface to XML Database

The XML interface to bdbxml is a component that will execute XML instructions against the database and then produce an XML result.

First, review the schema that will govern the interaction with bdbxml.

The schema consists of several elements. The operation element is the root node, containing the following attributes:

- **Database:** The full path to the database.
- **Type:** The operation type. Valid values are create, delete, update, inquire.
- **Docid:** The document ID for delete, update, or inquire operations.
- **xpath:** The XPath expression for inquire operations.
- **Wrapped:** "True" if you want the result wrapped in the Sleepycat Cocoon schema; "false" if you want just the XML document returned (the document that results from the operation – valid for inquire only).

The wrapped-false setting can be used to perform XSLT translations against the resulting document (versus the Sleepycat schema elements). The operation element can contain metadata elements and potentially one result element. The meta-

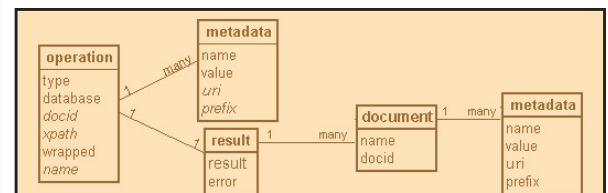


Figure 1 • Sleepycat schema

data elements tell the transformer which metadata items to extract from bdbxml for each document extracted.

The result element is a child of the operation node. It contains a result attribute that will contain “success” or “failure” depending on the result of the operation. It can also contain one or more document elements. The document element contains multiple occurrences of the metadata element and potentially text, as well.

The text would be the XML document retrieved from the database. The metadata element has a name and value attribute. One metadata element will be returned for each metadata element requested via the operation element.

Listing 1 is an example request document (Listings 1–4 and full source code are available at [www.sys-con.com/xml/sourcecc.cfm](http://www.sys-con.com/xml/sourcecc.cfm)).

This document instructs the transformer to inquire against the database and retrieve the document with the ID of 1.

The request document also asks that the following metadata items be retrieved: Name, Description, and Last Modified.

<C

```
<bdbxml:operation xmlns:sleepycat=
  "http://localhost:8888/sleepycat"
  database="/data/sample.dbXML"
  type="inquire" docid="1">
  <bdbxml:metadata name="name"/>
  <bdbxml:metadata name="
    description"/>
  <bdbxml:metadata name=
    "lastModified"/>
</sleepycat:operation>
```

### The SleepycatDBXMLTransformer

Take a look at the transformer that will process these instructions (see Listing 2).

The transformer parses the XML instructions document using the SAX methods – startElement and endElement. Once it has obtained the instructions, it executes the request. The results are then encoded into an XML response document (again, via SAX events), which is then passed back into the Cocoon pipeline for further processing.

The setup method obtains a reference to the SleepycatDBXMLManager stored in the session. The startElement and endElement methods employ the SAX interface to capture the XML instructions on the Sleepycat operation to execute (essentially the XML document discussed in the previous para-

graph).

When the startElement method detects the “bdbxml:operation” element, it examines its attributes to ensure correctness and then calls the startTextRecording method. This convenience method (implemented in AbstractSAXTransformer) will record the text of an element. The captured text is then obtained by calling the endTextRecording method, which is in the endElement method.

After the endElement method detects the end of the “bdbxml:operation” element, it calls the endTextRecording method to obtain the captured text. In this case, the captured text is the XML document that should be inserted into bdbxml (if performing an insert or update operation).

The performDbOperation method is then called by the endElement method. This method is responsible for executing the appropriate action based upon the XML instructions.

Lines 173–191 implement the delete functionality.

Using the XmlContainer and the ID passed in the XML, call the deleteById method on the container. This method deletes the document referenced by the ID from the container. The delete API call takes three parameters. The first is a DbTxn object if your work involves transactions. It didn't in this case, so we passed null. The second parameter is the XmlDocument, which was just created, and the third is an integer flag. The flag can be used to control fine-grained behavior, but we didn't need it so we passed a zero.

Lines 192–216 implement the create functionality. The create functionality employs a few simple bdbxml API calls.

First, a new XmlDocument instance is created, and the content object is set via the setContent method, passing in the XmlDocument text. The document is stored in the container via a call to putDocument, which takes three parameters (similar to the delete method previously described). The first parameter is a DbTxn object. Since you are not doing the work within a transaction, you default it to null.

The second parameter is the ID of the document to be deleted, and the third parameter is an integer flag. From a bdbxml perspective, those three API calls are all that is necessary to insert a document into the database (for persisting the XmlContainer, see the discussion that follows).

## Why Not a Full Content Management Solution?

We didn't want the overhead of a full content management solution. We wanted a lot of flexibility for developers without the “document management” overhead. We didn't want extensive versioning procedures. The users will manage the documents themselves. Nor did we want process or metadata overhead. No need existed to create extensive document-describing repositories for searching or to provide a consistent micromanaged process for document editing. Additionally, many content management solutions employ proprietary technologies that involve learning custom workflow languages. Cocoon and DB XML provide a simple approach that fits our needs. Cocoon provides great flexibility in the mechanisms employed to manipulate XML. A variety of technologies and tools can be brought to bear in the pipeline process, which makes

XML content manipulation both powerful and flexible. DB XML provides flexibility and extensibility by allowing metadata to be attached to documents and by providing an XML/XPath solution. Additionally, we can use XML documents to describe XML documents if we want to. The API is simple and provides round-tripping on the XML documents (which means they are guaranteed to look just as they did when they were put into the database).

Last (and perhaps most important), our solution had to be free. Since this tool is for internal use, we didn't want to spend much money on building, supporting, or maintaining the tool. If the tool is down for a period of time, we would have alternatives and workarounds available. The open source community provides one the best mechanisms for support of this nature.



Lines 262–291 implement the update functionality. The `XmlDocument` object to update is retrieved using its document ID from the database. The content is updated via the `setContent` method, and the `updateDocument` method is called with the `XmlDocument` object.

Lines 475–625 are responsible for creating the SAX events necessary to communicate the results of the operation.

Several lines of code are needed to generate the SAX events, but the code is simple and straightforward.

It's important to note that all events are passed through in the `startElement` and `endElement` methods by calling the `super.startElement` and `super.endElement` methods. Therefore, the resulting document from this transformer contains all of the input with the addition of the "Sleepycatresult" element. This design provides flexibility within the Cocoon environment because the full XML operation is available to manipulate in the stylesheets as necessary. Some additional overhead is involved, but for this situation, the flexibility was more advantageous.

In addition, the class isn't thread-safe. Implementing the `Recyclable` interface tells the Apache Avalon framework (which Cocoon leverages) that the component can be pooled. The Avalon framework creates an instance pool, permitting reuse of the component. It also protects against multiple pipeline threads entering the component by allowing access of only a single thread at any given time.

When it comes to persisting the database, this functionality is accomplished through the helper class `SleepycatDBXMLManager`. A reference to the `SleepycatDBXMLManager` was obtained in the `setup` method, or if the reference could not be obtained, then a new manager instance was built. The manager object, then, is stored by the session and exists for the life of the session. Look more closely at the manager object (see Listing 3).

The `SleepycatDBXMLManager` object is basically a `Hashtable` object that implements the `HttpSessionBindingListener` interface. (This changed to `HttpSessionListener` interface in the servlet 2.3 implementation. This interface is used because Cocoon is currently running under the servlet 2.2 implementation.) This interface allows the object to be notified anytime the session in which it is stored is

## What about the XML:DB API and the Cocoon Pseudo-Protocol?

The XML:DB API is a community effort to develop a standard programmatic interface to XML datastores. Cocoon supports the XML:DB API ([www.xmldb.org](http://www.xmldb.org)) through a pseudo-protocol; however, it currently supports only inquiry functionality (including XPath queries).

Our needs include an update interface. While an `XMLDBTransformer` is provided by Cocoon, it requires XUpdate capabilities and DB XML does not currently support XUpdate. In our case, XUpdate is largely overkill since we're concerned with documents, not document fragments.

Additionally, the small, consistent XML interface provided the simplicity we desired, allowing us to reuse our sitemap components and pipelines. Implementing separate inquiry and update interfaces seemed unnecessary, and our interface allows us to be more dynamic and flexible in how we create and execute instructions against the XML repository.

Finally, the XML:DB API provides no support for metadata. This lack of support may be intentional, as many suggest that you should use XML documents to describe XML documents. While this argument has some merit, this lack of support is inconvenient at best. DB XML provides metadata support, and we had every intention of using it.

The benefits of the XML interface and the flexibility of DB XML outweighed the detriment of being tied to one vendor's implementation (besides, we spent more time discussing the solution than actually implementing our simple transformer).

The XML:DB protocol is a solid protocol that will continue to see development and support within the XML community. If Sleepycat provided an XML:DB protocol or XUpdate support, then we would definitely revisit this decision since Cocoon and other products are embracing and extending their support of the XML:DB protocol.

destroyed. The session may be destroyed through either invalidation or timeout. In either case, the `SleepycatDBXMLManager` object receives notification of this event via the `unboundValue` method. This method retrieves the `XmlContainer` stored in the manager object and persists each container via a call to the `dump` method. The `dump` method is similar to the other API calls we have seen. The first parameter is the database name, which doubles as the filename.

The second parameter is an integer flag, which you default to zero, as it is not needed.

In this situation, load and dump are not necessary. You could allow `bdbxml` to handle the persistence through the normal open and close method calls. Using load and dump provides a simple way to persist the database into a single file, which makes the backup process easy. That's it from a transformer perspective. With just few lines of code, you have implemented a simple, reusable XML interface to `bdbxml`.

The new transformer can now be put to use.

*Note:* If you have problems with Cocoon finding the `bdbxml` libraries, make sure that the `bdbxml` bin and lib directories are in your `java.library.path`.

## Using the New Transformer Within Cocoon

Look at the Cocoon sitemap configuration to build the necessary pipeline (see Listing 4).

Five steps exist in the pipeline. First, the HTTP POST from the `TestTransformer.html` is encoded into a SAX document using the Cocoon-provided `RequestGenerator`.

Second, these XML events are then converted into the `bdbxml:operation` XML document using the default XSLT Transformer and the `convert.xsl` stylesheet. Refer to the sample code to see this simple stylesheet.

Third, the `SleepycatDBXMLTransformer` receives the operation instructions, executes them, and produces the XML reply, which is then serialized using the Cocoon-provided XML serializer. In a real application, this XML reply would be transformed into an HTML page (or the

like) that the user could use to view or edit the results.

You can compile the transformer by issuing the necessary Javac commands. Alternatively, a simple Ant script is provided for this purpose. A number of JARs from the Cocoon libraries are necessary for the compile to succeed, and the script properties must be modified to reflect the installation of Cocoon and bdbxml.

```
C<
ant compile
ant deploy
>
```

The first Ant target compiles the source and JARs it. The second Ant target moves the JARs, HTML, and XSL files into the WEB-INF directory (Cocoon Web-application root).

After successfully compiling and deploying the JAR file as well as the site map and the TestTransformer.html files, start the Cocoon server with:

```
<C cocoon.sh servlet
>
```

Point the browser at <http://localhost:8888/cocoon/TestTransformer>.

The TestTransformer pipeline is a simple interface to test the DBXMLTransformer.

Using this component within the Cocoon pipeline provides tremendous flexibility in putting together quick applications that need to be backed by a robust XML database.

The XML interface fits smoothly within the XML processing pipeline concept and allows extensive leverage of Cocoon's transformational capabilities. bdbxml is a key ingredient in the solution because it allows for simple integration into the application framework and provides superior performance. This application-specific nature coupled with the multiple-language APIs positions you well for future architectural directions.

### Conclusion

Businesses typically persist business data to an RDBMS for a variety of reasons. The RDBMS toolset is commonly available, an RDBMS environment is readily available in most enterprises, and the reporting/data extraction mechanisms are typically readily obtainable.

For these reasons and others, XML

databases have made more headway in other areas, such as caching, document-centric management (where RDBMS capabilities are not a good match), and the embedded toolset area (providing localized XML storage and retrieval support). Sleepycat's Berkeley DB XML addresses these areas well, providing an application-specific data storage solution with a simple API built on a robust and proven architecture. It is well suited to functioning as an XML storage and querying mechanism and provides powerful flexibility with a simple API.

We hope your awareness of application-specific XML databases has been raised and that this exposure will lead you to consider an XML database as a potential solution to your next XML storage problem.

An embedded XML database can immediately increase the options available to architects and developers. It also increases the flexibility and extensibility of a solution, and that leads to advantages in bringing new functionality to market sooner. ☼

DAN.HATFIELD@EDS.COM

## THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

Go to [www.SYS-CON.com](http://www.SYS-CON.com)



SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS. CHOOSE ONE OR TRY THEM ALL.

**FREE**  
**E-Newsletters**  
**SIGN UP**  
**TODAY!**

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at [www.sys-con.com](http://www.sys-con.com).

Exclusively from the World's Leading *i*-Technology Publisher

**SYS-CON MEDIA**



live wireless.

work wireless.

be wireless.

## CTIA WIRELESS 2004

is the one show where wireless standards are created and the technological direction of the industry is set. With the largest gathering of wireless engineers and technologists, this is where you will find the tools you need to help build and advance the wireless industry.

**Look at all CTIA WIRELESS 2004 has to offer the wireless engineer and technologist:**

- 6 CTIA educational sessions dedicated to exploring wireless technology
- IEEE Wireless Communications Network Conference (WCNC) 2004 – the industry's foremost conference for developing wireless standards and engineering
- WiFi Summit – the CTIA Smart Pass program, a cutting edge look at WiFi strategy and security
- A 400,000 square foot exhibit floor displaying the latest in wireless technology and applications

The most important  
technology event  
of the year!



welcome the wireless generation.

March 22-24, 2004 Georgia World Congress Center Atlanta, GA, USA [www.ctiashow.com](http://www.ctiashow.com)

John T. Chambers  
President & CEO  
Cisco Systems



Scott McNealy  
Chairman & CEO  
Sun Microsystems, Inc.



Russell Simmons  
Chairman & CEO, Rush Communications,  
Co-founder & Chairman, Def Jam Records



WRITTEN BY PHILIP BURTON  
& RUSSEL BRUHN



# Tree Structured Data and XML

## Visualizing and transforming data with SVG and XSLT

**X**ML is the appropriate format for semistructured data, that is, data with a natural tree structure. Trees are a special form of graphs, and a dialect of XML called GraphML now exists that provides a standard set of tags for describing them. To visualize or draw the graph we can use another XML dialect called SVG, or Scalable Vector Graphics.

SVG is a language for describing two-dimensional graphics and graphical applications and is a dialect of XML. In this article we discuss the representation of tree-structured data using XML and GraphML. We visualize the data using SVG and transform the data between the various XML documents using Extensible Stylesheet Language Transformations (XSLT).

### Introduction

A graph is simply a collection of nodes and edges and may be directed or undirected. In the case of a directed graph, an arrow connects the source to the target; for an undirected graph there is no arrow, only a line. A tree is a special case of a graph: there are no closed loops or circuits but all the nodes are connected. XML documents satisfy this definition and thus exhibit a natural tree structure. Examples of trees include decision trees, concept maps, and the structure of hydrocarbon molecules.

The structure of a tree is shown in Figure 1. The rectangles are the nodes and the edges are the lines connecting the nodes. Node A is a source and targets nodes B, C, and D; similarly, source node D targets nodes E and F.

The tree in Figure 1 can be represented in XML by a series of tags as shown in Figure 2. The nodes of the tree become tags in XML.

Alternatively, we can use a standard format like GraphML to illustrate a tree. In GraphML the tree shown in Figure 1 becomes the listing in Figure 3.

The "graph" element is the root element and the "edgedefault" attribute determines globally whether the graph is directed or undirected. Each edge element may contain an attribute called "directed" or "undirected" and has the values "true" or "false". Setting the attribute "directed" to "false" or setting the attribute "undirected" to "true" overrules the global setting.

The advantage of a standard format is that quite often a free viewer is available to render it in some fashion. If no free viewer is available, we may visualize the data by transforming it into SVG and using Internet Explorer. A free plug-in for SVG in Internet Explorer is available

from Adobe. Apart from the need to render a document or visualize it, you may still wish to transform a local XML document into another form for the purpose of exchanging it over the Web. This is best done in some accepted standard format, and XSLT is one way of accomplishing this. Christophe Jolif has discussed some alternative ways to make the XML-to-SVG transformation.

### Case Study: Subject Prerequisites

To illustrate, we'll use a mostly fictional example that could be taken from the subject catalog of any college or university. Subjects are generally taken in some set order; that is, the prerequisites must be done first. Listing 1 shows the prerequisites in XML format, as well as the post-requisites for each subject (Listings are available at [www.sys-con.com/xml/sourcec.cfm](http://www.sys-con.com/xml/sourcec.cfm)). The root element is <subjects>, and all the necessary information about each subject is included as a child element of it. What we have not shown is the schema in the form of a separate DTD or XSD document that details the rules for the tags. For example, in each <subject-prerequisite> element there can be only one <subject> child element and there would be limits on how many prerequisites and post-requisites each subject could have. Since the schema is not germane to our discussion, we have not included it.

By working XML we mean an XML document that conforms to a schema that is being used locally within a particular organization. A local dialect is not necessarily used outside the organization. To exchange data like this with another university, it would be advantageous if a common data format existed. This could take the form of a shared dialect with a publicly available and

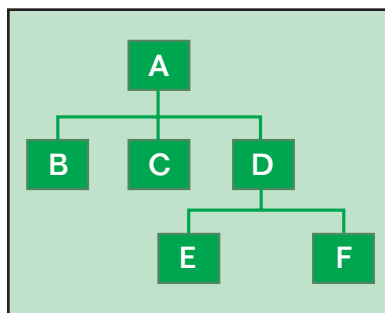


Figure 1 • Tree

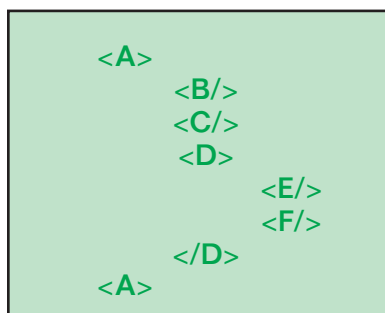


Figure 2 • XML

### AUTHOR BIO

Philip Burton earned his Ph.D. in Mathematical Physics from the University of Queensland in 1996. He is an assistant professor in the Department of Information Science at the University of Arkansas at Little Rock. Philip is a member of the Institute of Electrical and Electronic Engineering (IEEE) Society, the American Mathematics Society (AMS), and the American Physical Society (APS).

Russel Bruhn earned his PhD in electrical engineering from Washington State University in 1997. He is an associate professor and chair of the Department of Information Science at the University of Arkansas at Little Rock. His research interests are in the areas of creating innovative curriculum, computers and applications of XML with SVG graphics.



# 2004 Web Services on Wall Street®

Show and Conference

February 3-4, 2004 (Tues-Weds)

Metropolitan Pavilion, NYC

125 West 18 St at 7th Ave, New York, NY

**The 4th Annual Wall Street event for Web Services, .NET, Java, XML and other Web Services technology for the global financial markets.**

Register and Save \$100. Full program \$295 for 2 days of Web Services knowhow.



Register now and save \$100. Full 2-day conference program - \$295. \$395 after the deadline.

This is the lowest cost for any major financial program, and it's in New York City.

You are cordially invited to join us for the 2004 4th Annual Web Services on Wall Street conference.

This is your best investment for two days of top Wall Street speakers, Web Services case studies, major programs by Computer Associates, Sun Microsystems, IBM, Microsoft, among other leading providers of Web Services, two industry luncheons, an opening networking reception, and a major exhibition with leading Web Services vendors.

## Show Schedule

**Free Show:** Tues, Feb 3 4 - 6 pm  
Weds, Feb 4 10 am - 4 pm

**Conference:** Tues, Feb 3 9 am - 4 pm  
Weds, Feb 4 10 am - 4 pm

Come explore a Web Services supermarket focused exclusively on Wall Street and the global financial markets.

If you don't have time, come for the FREE Show.

Register online at:

[webservicesonwallstreet.com](http://webservicesonwallstreet.com)

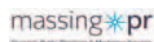
Thank you to our media and industry sponsors:



Conference Producer



PR Partner



For more information on the event, to be a sponsor or exhibitor, contact:  
Russell Flagg, Flagg Management Inc  
(212) 286 0333, [flaggmgt@msn.com](mailto:flaggmgt@msn.com).

For information on the conference, to  
Pete Harris, Lighthouse Partners,  
(718) 237 2796, [pete@lighthouse-partners.com](mailto:pete@lighthouse-partners.com)



fixed schema.

## Transformation to a Standard Format

What we are describing is a two-step process for making the transformation from the input XML document to the output SVG document. The intermediate format we are using is GraphML, which is a potential industry standard. There is another advantage to doing things this way: the overall XSLT transformation is simplified if it is broken down into two transformations. It would be hard to find a simpler XML dialect than GraphML, and its use as the intermediate considerably simplifies the XPath expressions used. These can become quite complicated; indeed XPath is generally considered one of the more difficult aspects of XML and any scheme to simplify its use is a plus.

The transformation from the working XML document, Prerequisites.xml, is accomplished by the XSLT Stylesheet XMLtoGrML.xslt, shown in Listing 2.

## GraphML

The application of the stylesheet XMLtoGrML.xslt on the input XML file produces the GraphML document, GrML.xml, shown in Listing 3. A GraphML document may contain any number of <graph> elements. Each must include a mandatory "edgedefault" attribute that determines globally whether the graph is directed or undirected. A <graph> element can contain <node> and <edge> elements in any order.

GraphML was designed to represent both basic and specialized graphs. A specialized graph would require additional information in the form of data elements that describe layout and graphics. GraphML consists of a core component that describes the basic graph structure. The main elements of it are shown in Figure 3. The extension mechanism allows additional data specific to one graph to be included. In this article we will make use of the simpler aspects of GraphML.

## Transformation to SVG

Both GraphML and SVG are standard file formats. A general-purpose XSLT stylesheet can now be written that will transform the GraphML into SVG. This is another advantage of using a two-step process. The last step involves the design of a stylesheet that need not be altered. Rather than reinvent the wheel, we have chosen to adapt an excellent

```
<graph edgedefault="directed">
  <node id="A"/>
  <node id="B"/>
  <node id="C"/>
  <node id="D"/>
  <node id="E"/>
  <node id="F"/>
  <edge source="A" target="B"/>
  <edge source="A" target="C"/>
  <edge source="A" target="D"/>
  <edge source="D" target="E"/>
  <edge source="D" target="F"/>
</graph>
```

Figure 3 • GraphML

XSLT stylesheet program written by Christophe Jolif, shown in Listing 4. This is another good reason for working with standard formats: chances are that somebody has already written a good program for you.

## Scalable Vector Graphics

Scalable Vector Graphics is a language for defining graphics; its purpose is to define visual appearance. SVG was created by the W3C to handle vector

ical statements to describe the image. Due to this vector nature, SVG files can be quite small in size and are easily transported over the Internet. After downloading and expansion the image remains the same in size since the mathematical formulas describing the image do not alter.

SVG works hand-in-hand with other technologies. SVG can exist on its own, be used within an XML document, be referenced from HTML, display JPEG images, and can also utilize JavaScript and Cascading Style Sheets (CSS). With CSS you can extend SVG's capabilities. Every line, shape, and style property can be controlled and customized. Thus with SVG we can personalize our graphics.

The XSLT program of Listing 4 acts on the GraphML code in Listing 3 to produce the output SVG code of Listing 5. Line 1 is the standard XML declaration. Line 2 shows that the size of the SVG graph produced has been set to have a width of 8 inches and height of 11 inch-

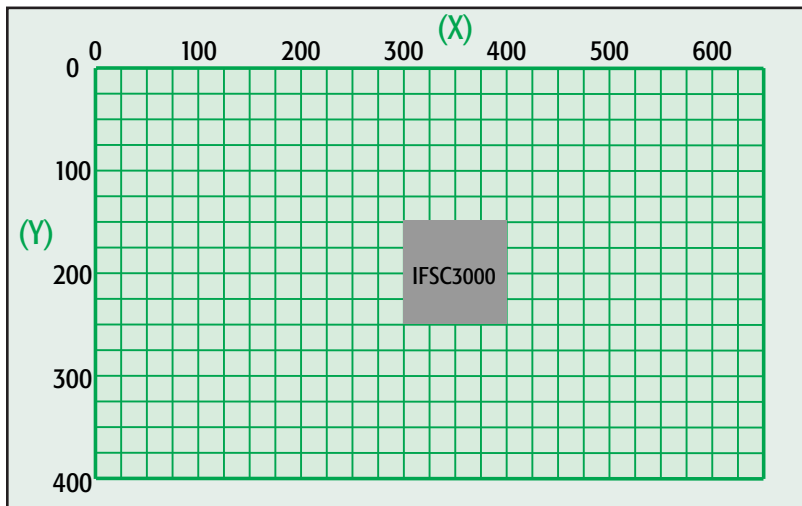


Figure 4 • Standard SVG document coordinate system

graphic display and animation in XML. SVG is entirely text based and is editable in text editors. It is easily human readable; for example, a rectangle located at position (450,150) with a width of 100 pixels and a height of 100 pixels is described in code as:

```
<rect x="450" y="150" width="100"
      height="100">
```

JPEG images and GIF files are raster graphics that use bitmaps. Because of this, when a JPEG image or GIF graphic is enlarged the picture becomes fuzzy. SVG, on the other hand, is resolution independent because it uses mathemat-

es. This ensures that the SVG graph will fit on a legal-size page.

Applying style to an object, or grouping of objects, is handled by the <g></g> element. This gives developers the option of applying a particular style to an element or associating elements together. Listing 5 shows how to individually apply a style to each rectangle. Figure 4 shows the X-Y coordinate system of a standard SVG document. The default unit of measurement is the pixel and is what we use here. Other popular formats are inches (in), centimeters (cm), and points (pt).

The following code produces the rectangle shown in Figure 4.

```
<rect x="300" y="150" width="100"
  height="100" fill="lightgrey"/>
<text text-anchor="middle" x="350"
  y="205">IFSC3000</text>
```

The top left-hand corner of the rectangle is the point (300,150). The width and height of the rectangle are 100 pixels, and the center of the text "IFSC3000" is anchored to the point (350, 205). We could have chosen the type of font for the text using CSS, but we used the default setting of our XML processor which was XMLSPY.

The XSLT program in Listing 4 transforms the standard GraphML code in Listing 3 into the SVG graph in Figure 5. By recursions through the "node" and "edge" elements of the GraphML program a series of rectangles and lines connecting the rectangles is produced. When the text code in Listing 5 is viewed in the IE browser the SVG graph in Figure 5 is the result. The text code in Listing 5 is a series of rectangles, like the one produced in Figure 4, connected by lines that are created by the line element:

```
<line x1="250" y1="100" x2="350"
  y2="150" style="stroke:black"/>
```

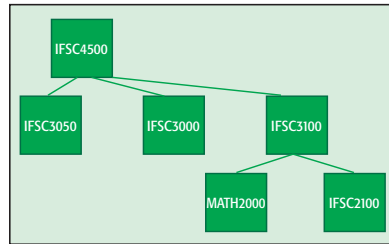



Figure 5 • SVG graph

Once again, the coordinate system is in pixels and is exactly the same as in Figure 4. The above line starts at the point (250,100) and ends at the point (350,150). This line in Figure 5 connects the IFSC4500 rectangle with the IFSC3000 rectangle. The line style stroke is black.

### Summary

GraphML is a comprehensive and easy-to-use file format for graphs. Its simple structure makes it convenient to use as an intermediary format between a working XML document and an SVG output document. Both GraphML and SVG are standard file formats, which confers stability on the XSLT stylesheets used in the second step of the transformation. Use of a simple XML dialect like

GraphML also confers the advantage of simpler XPath expressions and a simplified design process for the XSLT. Finally, GraphML is a candidate industry standard allowing the intermediary document to be used for data interchange over the Web. 

### References

- Jolif, Christophe. "Comparison between XML to SVG Transformation Mechanisms." [www.svgopen.org/2003/papers/ComparisonXML2SVGTransformation-Mechanisms/](http://www.svgopen.org/2003/papers/ComparisonXML2SVGTransformation-Mechanisms/)
- Scalable Vector Graphics: [www.w3.org/Graphics/SVG/](http://www.w3.org/Graphics/SVG/)
- Adobe downloads: [www.adobe.com/support/downloads/main.html](http://www.adobe.com/support/downloads/main.html)
- Gardner, John and Rendon, Zarella. *XSLT & XPATH: A Guide to XML Transformations*. Prentice Hall, 2002.
- Cagle, Kurt. *SVG Programming: The Graphical Web*. Springer Verlag, 2002.
- Watt, Andrew; Lilley, Chris; et al. *SVG Unleashed*. Sams Publishing, 2003.

PJBURTON@UALR.EDU

REBRUHN@UALR.EDU



### FEATURE

#### Community Integration and XML

Extending integration between trading communities



### CASE STUDY

#### First Class Services for the

Travel Industry How Agent Ware manages the quality of its Web services system



### BOOK EXCERPT

#### Mastering XML: Java Programming with XML, XMI and UML

Long-lived loosely coupled asynchronous transactions



### HOW-TO

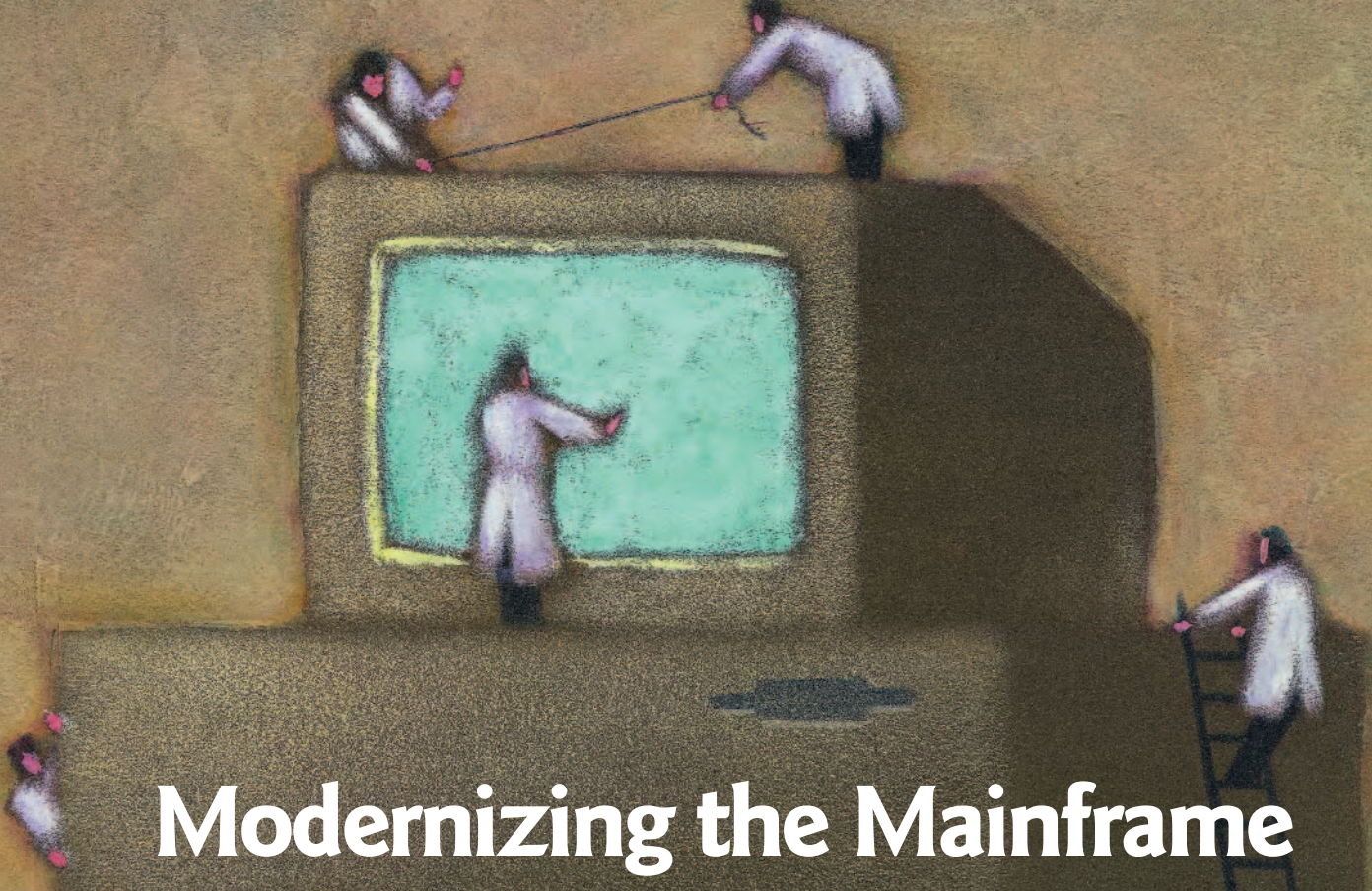
#### Rendering a Connected Tree

XSLT makes it quick and painless

# DON'T MISS XML-J FEBRUARY

## REAL-WORLD SOLUTIONS





# Modernizing the Mainframe

WRITTEN BY  
JOE GENTRY

*Unleashing the power of XML and Web services*

Central Hudson Gas and Electric, a New York State utility company, wanted to find a way to improve its customer service by creating a Web-based platform where customers could view and retrieve bills online. Replacing or rewriting the company's 20-year old mainframe billing application wasn't an option. The mainframe worked great and could satisfy the company's growth plans for some time to come.

Instead, Central Hudson took the simpler route of deploying an XML integration server between its Customer Information Service system and an Internet portal. As a result of this very simple project, Central Hudson estimates it can generate yearly savings of \$500,000 by offloading customer calls to a self-service Web site.

Central Hudson is not alone in taking a second look at its admittedly "unsexy" mainframe and finding impressive new sources of productivity and cost savings.

While many signs point to an economic recovery in 2004, it is clear that chief executives are not prepared to return to the free-spending ways of the late 1990s. Chief information officers are being told they will simply have to do more with the same or fewer resources. IT departments that have been trimmed in recent years through downsizing or outsourcing will be under intense pressure to meet new business-driven goals without double-digit increases to budgets.

The end result is that organizations will increasingly be forced to reevaluate the role and value of the mainframe in their operations. Business goals will dictate that information and resources kept on the mainframe must be made available to a much broader audience within and outside the corporation. The crucial question is how best to achieve those future e-business goals within the time and budget constraints set out by management.

For a great many organizations, the answer will be to implement standards-based technologies that will modernize the mainframe. By unlocking the power of technologies such as XML and Web services in the enterprise, companies will not only be able to mine the decades of experience invested in developing legacy applications, they will also be able to exploit a highly stable and powerful platform to meet their future business needs.

## Evaluating the Strategies for Modernizing

Organizations will typically choose among four strategies for modernizing their mainframe applications. The first and least-dramatic strategy is to simply make sure that the mainframe is optimized as much as possible. This might include evaluating all databases and applications currently deployed in production to ensure that they are making the best use of resources, as well as providing developers and administrators with visual development platforms to boost productivity.

The second strategy is to keep the business logic of the applications on the mainframe, while providing the user with a more "friendly" interface to the mainframe data such as a Web browser or mobile device. The third strategy is to actually integrate the mainframe applications with new technologies, while the applications themselves, as well as the data, still reside on the mainframe. Finally, the most dramatic approach (short of replacing or rewriting the applications, which is outside of the realm of "modernizing") is to migrate the mainframe application source code to a different platform with little or no change. The data might also be migrated, or might remain on the mainframe.

In the current business climate, many organizations have already come to the conclusion that reusing existing mainframe applications by implementing one of the four strategies mentioned makes the most sense when compared to the time

and labor costs involved in replacing or rewriting the application.

After all, mainframe legacy applications are usually still in place because they are critical to the enterprise. They hold strategic data on customers, operations, and financial reporting. They often handle transactions that invoke multiple databases and systems or contain custom coding that has been developed and fine-tuned over long periods of time.

Beyond the strategic role played by mainframe systems, organizations also recognize that these systems excel at high-speed transactions, handling, and reliability. As such, they are ideal back-end engines for new Web-facing e-commerce applications. Add to these two factors the current competitive pressures that demand new functionalities be brought to market in months – not years – and it becomes apparent that leveraging mainframe systems may be the only effective way of meeting cost, time, and feature requirements.

### Customer Access and HIPAA Compliance

Consider the situation faced by American Fidelity Assurance Company, a leading provider of insurance products to the education sector and trade associations. The Oklahoma City-based company was under pressure on two fronts to find solutions to modernize its mainframe applications. On the first front, there was a clear business need to provide its customers with a Web portal to gain access to information and resources kept on the mainframe that were previously only available to American Fidelity employees. The second front involved meeting a federal mandate under the Health Insurance Portability and Accountability Act of 1996 for the exchange of electronic health records between health-care providers.

James Lupton, vice president in Information Systems for American Fidelity, says the company initially looked at rewriting or replacing many of its mainframe applications to accomplish the two goals, but after evaluating the options, it was clear that this route made no business sense. "It would have cost us seven figures to complete and taken several years to do it," he says. "Instead, we were able to leverage our existing applications at less than 20 percent of what it would have cost us to rewrite them."

For the first project, developing a Web portal for American Fidelity's six different customer groups, the solution was to deploy a message-oriented middleware product (Software AG's EntireX). Such systems act as brokers between a wide range of legacy applications and newer Internet technologies. They enable the placement of an application on the platform that fits it best, and then facilitate access to the data from wherever it resides. In this case, the portal application was deployed on network servers, mainframe applications were executed, and data was retrieved from the mainframe databases. Some of the mainframe applications generated an XML Data Stream to pass down to the Web applications to render as a PDF or other Internet-displayable format.

Using this technology, American Fidelity was able to develop its AFAdvantage portal in just seven months. Today, more than 11,000 customers – as well as 1,200 brokers, 400 account managers, and 260 bookkeepers or group administrators – are plugged into the portal on a daily basis. Data access takes place in real time and response time handily beat the sub-four-second goal set by the deployment team. Lupton says the system will have no problem scaling for future growth.

Based upon one simple cost-saving factor, a reduction in the number of calls received for insurance forms, the Web portal is saving American Fidelity at least \$495,000 a year – twice the initial deployment cost.

On the second project – meeting HIPAA compliance – American Fidelity once again exploited the power of XML. It implemented message-oriented middleware to accept HIPAA transactions and convert them into an XML format. Those documents are now being stored in an XML database, and can be searched, retrieved, and delivered to a variety of systems in all required formats.

"When we started down this path we had a consultant come in and tell us, 'You're behind the curve, you're mainframe based and your systems can't talk to one another,'" says Lupton. "While that was never actually the case, we have certainly proven his assessment false. We now have a system in place that gives us the confidence to move ahead with a number of other projects."

In fact, American Fidelity has 11 initiatives on the table for 2004 to further integrate its mainframe and networking infrastructures.

### Technology Drivers for Modernizing

Drivers for modernizing the mainframe vary depending on the individual organization and its business goals. These drivers may range from a desire to simply represent the information from a mainframe green screen on a Web browser, to a need to access the legacy application code via the Internet, to a requirement to build true interoperability across databases and applications (including the need to persist, monitor, and audit data as it moves around the enterprise).

Whatever the driver for modernization, organizations are finding that XML, used in conjunction with a Web services-based architecture, is an excellent foundation for extending the life, functionality, and value of mainframe systems. By allowing data to be dynamically repurposed into multiple presentation formats, the technology provides a means to deliver standard access to both structured and unstructured information. Furthermore, the implementation of a Web services-based architecture enables universal access to all information systems.

This is not to imply that XML and Web services are required in all cases. In many instances, a simple 3270 screen-scraper system may be all that is needed to accomplish an organization's goals. The difference today is that screen-scraper technologies can be combined with XML technology to quickly and inexpensively create easy-to-use Web interfaces. The underlying Natural or COBOL code has not been changed, but the user experience has been modernized. This approach can extend the life of a wide range of legacy systems with minimum effort.

### Give the People What They Want

Perhaps the biggest driver facing companies and organizations is the need to access legacy applications from Internet devices or desktop systems. Today, organizations want to reach out to their customers, suppliers, and business partners in new ways through Web applications and business intelligence tools. In addition, companies are looking to provide wider access to legacy applications and data throughout their own organizations. XML integration servers are providing the answer.

Consider this challenge faced by the New York City Department of Buildings. In the past, information in the agency's database could be accessed only by city employees. This

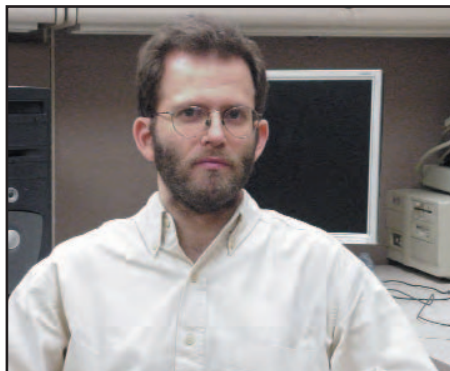


• Jim Lupton, Vice President in Information Systems, American Fidelity



meant that citizens, contractors, and builders had to line up in borough offices to check the status of a building permit or to look up information on a building's history, such as violations or complaints. The information in the system was public data, and could be made available via the Internet. The problem was finding a way to pull the information out of the mainframe system and publish it in an easy-to-use format.

The underlying system was more than a decade old, and the cost to rewrite the application simply wasn't an option, says David Presley, assistant director of application development. Instead, the Department of Buildings deployed mes-



• David Presley, Assistant Director of Application Development, New York City Department of Buildings

sage-oriented middleware and built an Internet window into the millions of pieces of information on the 900,000-odd buildings in all five New York boroughs. The system puts a software-based "wrapper" around data and programs written in Software AG's Natural programming language. This allowed the mainframe data to be passed to a Java-based Web application.

The project took less than a year to complete and has been embraced by both the building community and New York residents. The system now receives about 200,000 requests for informa-

tion each day, and provides access around the clock.

"Everyone's seeing benefits as a result of this project," adds Presley. "It's saving us money because we don't have as many people standing in line, it's saving the building community time and money, and it's democratized the information in our system."

Remarkably, the application development team has built more than 80 servlets tapping into the department's legacy systems, and Presley says many more are on the way.

The State of Minnesota faced a similar challenge when it undertook a project to provide access to the state's welfare system via the Web. The state's mainframe-based MAXIS welfare information system was previously only accessible via green screen terminals. The state wanted to make elements of the system accessible via the Internet, and to be able to use the mainframe application in new ways via a Web services infrastructure. By first mining the code, it was able to break the application down into reusable components. The technology team then deployed middleware to serve as the interface between MAXIS and the Web applications.

Some 30,000 employees, providers, and clients now have access to a universal and customizable system, to provide social service benefits to any recipient. Notably, the State estimates it saved \$13 million by not having to rewrite the application.

## SOA: Providing the Keys to the IT Shop

There is a fundamental problem that applies to most organizations. Much of the value that exists in an enterprise's IT systems is locked away or not accessible to those who could benefit from it because access to data (and the tools used to analyze and exploit the data) is usually restricted to specific user communities. This is particularly true in the mainframe world.

A service-oriented architecture (SOA) provides the means to begin unlocking the vast amount of value tucked away in an organization's mainframe infrastructure. At its heart, it offers the capability to develop a collection of services that may communicate with one another. Such an architecture can be used to divide larger applications into discrete modules and make those modules available to other applications. This process addresses one of the key modernization challenges, which is to build interoperability between databases and applications.


SOAs are certainly not new, but a wide range of new technologies based on Internet standards have made the concept much more technologically and economically feasible. Technologies such as Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) provide a simple yet effective way to exchange XML documents and describe how to invoke a Web service. Standards such as Universal Description, Discovery, and Integration (UDDI) provide a way to register Web services so that one Web service application may locate and invoke another.

As organizations develop SOAs, they will be able to start tackling large organizational challenges, such as providing various business units with a customized "single view" of all the relevant data that resides throughout the enterprise, or supporting the ability to persist, monitor, and audit data as it moves around the enterprise from system to system.

XML-based products have been developed to orchestrate the aggregation, transformation, and filtering of XML messages between multiple back- and front-end systems and Web services. XML database servers now provide the means to store, monitor, and audit data as it moves around the enterprise. Think of it as a completed circle. An XML integration server, along with XML connectors, handles the interface with legacy and newer enterprise systems; XML brokers orchestrate the intelligent routing and delivery; and native XML servers complete the circle by providing persistence, historical analysis, and full auditing. Together these components make it possible to deliver an SOA that truly modernizes the mainframe.

## Long-Term Value for the Organization

At the end of the day, organizations are looking for a way to better protect the investment and the value of their IT assets. The good news is that technologies, to a great degree based on XML and Web services, have provided that means. In fact, mainframe systems in the years ahead will serve as the backbone for a wide range of exciting new service-based applications.

Central Hudson, American Fidelity Assurance, the State of Minnesota, and the New York City Department of Buildings didn't just prolong the life of their mainframe systems – they used new technologies such as XML and Java in concert with legacy applications to make a big contribution to the company's bottom line. Looking at the year ahead, that will be a key factor driving organizations of all shapes and sizes down the same path toward modernizing their mainframe infrastructure. 

### AUTHOR BIO

*Joe Gentry has more than 19 years of experience in strategic marketing, product management, and software development. As senior director of product marketing for Software AG, Inc., he is responsible for overseeing product definition, solutions planning, positioning, and market launch. Joe has managed software products through all phases of the life cycle and has become a leader when launching products into the marketplace.*

JOE.GENTRY@SOFTWAREAGUSA.COM



1





































# 16





## The Leading Magazine for Enterprise and IT Management



# LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* does not feature low-level code snippets but focuses instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

### Regular features include:

- Advice on Linux Infrastructure
- Detailed Software Reviews
- Migration Advice
- Hardware Advice
- CEO Guest Editorials
- Recruiting/Certification Advice
- Latest News That Matters
- Case Studies

# SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

# \$49<sup>99</sup>

12 ISSUES/YR

\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

## SUBSCRIBE TODAY!

[WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR

VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)



WRITTEN BY ANDREW SOLYMOSI

# Processing XML with C# and .NET

## A solution that's simpler than you might expect

Microsoft's counterpiece to Java, the new C# programming language with its rich .NET library, uses XML as a core technology. This article presents some basic ideas, for example creating and manipulating a DOM tree, and reading and writing XML streams.

I also compare .NET's solution with the SAX model, and finally I show how a complex XSLT algorithm can be more simply implemented in C#. The source code can be downloaded from [www.sys-con.com/xml/sourcec.cfm](http://www.sys-con.com/xml/sourcec.cfm).

### XML Processors

To process an XML document means to extract information from it. Often the extracted information should be output to a new XML (or perhaps HTML) document that's similar to the original one – then we address the transformation.

Which processor to use for a given task is not a trivial question. There are a number of ready-to-use XML processors on the market – like Cocoon and Axxit – and any XML-enabled browser (like Netscape Navigator 6 or Microsoft Internet Explorer 6.0) has a built-in XML processor. They require processing instructions, which can be contained in an XSLT document for complex operations. XSLT is, however, a special style and not a very convenient programming language. Many would prefer to fiddle around with well-known, conventional programming structures from C++ or Java. For them, a real alternative is to write their own XML processor. Modern programming languages like C# and Java offer great support for this task in the form of class libraries; in the case of C#, these are .NET classes.

### Processing Methods

There are two ways to process an XML document: online and offline. Offline processing means not being connected to the XML source, so the document has to be loaded (typically as a DOM tree) into memory beforehand. In the space-time trade, you lose space (memory) but gain time (speed) for the processing. This is the best method if most parts of the document are going to be processed, especially if they're going to be processed repeatedly.

Online processing (reading the document piece by piece) can be slow, but it works with less memory. It is advantageous if only certain parts of the document will be processed, or if the processing is sequentially straightforward.

C#'s built-in .NET library provides all the classes necessary for processing an XML document both ways. They are placed in the System.Xml namespace (with its nested namespaces System.Xml.Schema, System.Xml.Xsl, etc.).

The abstract classes XmlReader/XmlWriter provide the basis for online processing; XmlReader represents a fast, read-only forward cursor in an XML stream, while XmlWriter provides an interface for producing XML streams. The basis for offline processing is the class System.Xml.XmlDocument (with its superclass XmlNode) representing an XML document as a DOM tree (i.e., memory intensive). We'll investigate this option first.

### Creating an XML Document

The XmlDocument constructor (usually called without parameters) creates an empty in-memory XML document. Its methods, like InsertBefore, InsertAfter, and AppendChild, build the DOM tree in memory. They need Xml-

Node parameters, which can be created by the XmlDocument's method CreateXxx, where Xxx stands for Element, Attribute, Node, Comment, ProcessingInstruction, CDataSection, DocumentFragment, XmlDeclaration, DocumentType, or EntityReference. They create a node in the context of the document but don't attach it to the tree:

```
XmlDocument doc = new XmlDocument();
// empty
XmlElement elem = doc.CreateElement
("NewAndOnlyElement"); // XmlElement
is a special XmlNode
elem.InnerText = "Data of the element";
... // many other properties to
configure the element
doc.AppendChild(elem);
// appends node to document
doc.Save("data.xml");
// save the document to a file
```

The content of the text file data.xml (after putting this program segment into a Main method and running it) is:

```
<NewAndOnlyElement>Data of the element
</NewAndOnlyElement>
```

In this way an XML tree of any complexity can be built step by step:

```
XmlNode node = doc.CreateComment
("This is a comment"); // XmlComment
is subclass of XmlNode
doc.AppendChild(node);
node = doc.CreateProcessingInstruction
("xml-stylesheet", // target
"type='text/xsl' href=data.xsl"); //
data
doc.AppendChild(node);
node = doc.CreateCDataSection("<p>");
...
```

Since XmlDocument's AppendChild method is inherited from XmlNode,

#### AUTHOR BIO

Andrew Solymosi graduated from the Leningrad State University, Soviet Union, and earned his PhD in computer science at the Erlangen University, Germany. He teaches programming languages and software engineering at the Technical University for Applied Sciences, Berlin, and is working on an XML portal project in Orlando, Florida.

nodes can be appended not only to doc but also to any of the created nodes.

## Navigating in the DOM Tree

After building (or loading) an XML tree, you can navigate over it and manipulate it with XmlDocument's properties (most of them are inherited from XmlNode). They have intelligible names:

- DocumentElement delivers the root XmlElement of the tree.
- Indexer [ ] delivers the XmlElement with the specified index.
- Attributes delivers an XmlAttributeCollection object, a collection of XmlAttribute (a special Xml-Node) objects.
- ParentNode, PreviousSibling, NextSibling, FirstChild, and LastChild navigate over the neighboring XmlNode objects for further manipulation or evaluation.
- ChildNodes delivers all nodes as XmlNodeList, a collection of XmlNode objects – it can be read, for example, with foreach.
- NamespaceURI, Prefix, BaseURI, LocalName, and (qualified) Name deliver string objects about the document's URI.
- OuterXml delivers a string containing the markup (XML text) of the node (with all its children).
- InnerXml gets or sets a string containing the markup of all the children.
- InnerText gets or sets a string, the concatenated text values (without XML markups) of the node and all its children.
- PreserveWhitespace (with set), HasChildNodes, and IsReadOnly are bool properties.

## Reading XML Data

There are additional ways to fill an XmlDocument object with data. The most important is its Load method with a string or stream, or TextReader or XmlReader parameter. It assumes that the XML document exists in text form in a stream, e.g., in a text file. The string-Parameter can be any URI. The difference between the overloaded versions is the navigation over the document before loading (because there is no need to load the whole document, only parts of it). Stream or Text-Reader allows sequential navigation; XmlReader can navigate to any component of the XML tree before reading it (see online processing in the next section).

If we have the XML data as a string object, we can use the LoadXML method:

```
const string xmlData =
    "<family location='Orlando'
    class='middle'>" +
    "    <lastname>Solymosi</lastname>" +
    "    <father>Andrew</father>" +
    "</family>";
doc.LoadXml(xmlData);
doc.Load(new StringReader(xmlData));
// equivalent alternative
```

Most void methods of XmlDocument (many inherited from XmlNode) suite to read and write the content of the DOM tree to and from a stream.

- Load (with parameter Stream or string or TextReader or XmlReader) and LoadXml(string) fill an XmlDocument object with data.
- Save (with parameter Stream or string or TextWriter or XmlWriter) writes the content of an XmlDocument object to a stream.
- WriteTo writes OuterXML (markup of the root node); Write-Content-To writes InnerXML (the markup of all children) to an XmlWriter stream.
- Normalize puts the tree into a "normal" form in which only markup separates XmlText nodes (i.e., there are no adjacent XmlText nodes).
- RemoveChild and RemoveAll delete child element(s).

## Online Processing

The abstract classes XmlReader and XmlWriter define an infrastructure for online processing, i.e., without building the DOM tree in memory. XmlReader is, however, not an implementation of the SAX model, rather a compromise between DOM (with the simple programming interface) and SAX (non-cached forward-only reader allowing the user to skip data). While a SAX reader activates the application's callback methods (push model), the methods of an XmlReader class must be called by the application (pull model). Some of the advantages of the latter are:

- SAX requires the user to build complex state machines. A client for XmlReader builds a top-down procedural refinement.
- XmlReader allows the client to read multiple streams (e.g., for aggregation); with SAX it's hard work.
- An XmlReader application can be built on top of the SAX model (layering).
- SAX copies the data from the parser buffer into the client's string object; XmlReader can use the client's string parameter as parser buffer so a string copy can be avoided.

- SAX calls the client for each item (including attributes, processing instructions, and white space), while XmlReader's client can skip items (selective processing).

XmlReader has three implementations in .NET 1.1: XmlTextReader (this fast reader checks only well formedness and throws XmlException on failure), XmlValidatingReader (additionally validates against a DTD or a schema), and XmlNodeReader (for reading a DOM subtree). XmlWriter has only one implementation in .NET 1.1: XmlTextWriter. The counterpiece to XmlNodeReader (the class XmlNodeWriter) has not yet been implemented. However, people are welcome to write their own custom XmlReader and XmlWriter, extending the standard's functionality.

The following method shows how XmlReader moves along the document stream and displays the name of each element:

```
static void ReadMyDocument
(XmlReader reader) {
    while (reader.Read()) {
        if (reader.NodeType ==
            XmlNodeType.Element ||
            reader.NodeType ==
            XmlNodeType.EndElement)
            Console.Write (
                (reader.NodeType ==
                    XmlNodeType.EndElement ?
                        "/" : "") + reader.Name + " ");
    }
}
```

This method can be called with an XmlReader implementation as argument, e.g., with XmlText-Reader:

```
public static void Main
(String[] args) {
    XmlReader r = new
        XmlTextReader(args[0]); // URL
    ReadMyDocument(r);
}
```

If the command-line parameter contained the name of a text file with the content

```
<family location='Orlando'
    class='middle'>
    <lastname>Solymosi</lastname>
    <father>Andrew</father>
</family>
```

the output would be

```
family lastname /lastname father
/father /family
```



This shows how `XmlReader` sequentially goes through the input document. A stop can be made at a node, which can be inspected with `XmlReader`'s methods and properties:

```
string qName = reader.Name;
string localName = reader.LocalName;
string namespaceURI =
    reader.NamespaceURI;
XmlNodeType nodeType =
    reader.NodeType;
string value = reader.Value;
bool hasAttributes =
    reader.HasAttributes;
int numberOfAttributes =
    reader.AttributeCount;
...
```

`XmlReader` properties (all read-only, i.e., without set) deliver information about the current node. The most important are:

- `Eof`, `HasAttributes`, `HasValue`, `IsDefault`, `IsEmptyElement` deliver bool.
- `AttributeCount` and `Depth` deliver int.
- `BaseURI`, `indexer [ ]`, `LocalName`, `Name`, `NamespaceURI`, `Prefix`, `Value` and `XmlLang` deliver string.
- `NodeType` delivers an `XmlNodeType` enumeration value: one of `Document`, `Text`, `XmlDeclaration`, `Element`, `EndElement`, `Entity`, `EndEntity`, `EntityReference`, `Attribute`, `Comment`, `ProcessingInstruction`, `CDATA`, `DocumentFragment`, `DocumentType`, `Notation`, `White-space`, `Significant-Whitespace`, or `None`. It can be used in a switch.

`XmlReader`'s void methods navigate forward alongside the document:

- `Read` delivers the bool value true if the next node is read successfully and throws `XmlException` if it detects an error in well formedness.
- `ReadXxx` reads the next required node (otherwise throws `XmlException`) with `Xxx = AttributeValue` (with parameter `Text`, `EntityReference` or `EndElement`) `ElementString` (simple text-only element), `StartElement`, `EndElement`, `InnerXml`, `OuterXml`, or `String` (contents of an element or text node).
- `MoveToXxx` (with `Xxx = Content`, `Element`, `Attribute`, `FirstAttribute` or `NextAttribute`) moves to the required node and skips everything else.

The `ReadXxx` and `MoveXxx` methods perform a depth-first traversal of the tree, i.e., in the order that the sequential document stores it. The classic method

for processing a document is to read it in a while cycle and switch according to `XmlNodeType` (see Listing 1).

As you can see from Listing 1, `XmlReader` requires knowledge about the current node type before calling the appropriate method, and SAX calls the appropriate method, and according to the current node type.

## Navigation

`XmlNode` provides methods for navigating the DOM tree horizontally and vertically to the neighboring nodes (like `ChildNodes`, `FirstChild`, `LastChild`, `ParentNode`, `NextSibling`, and `PreviousSibling`). Jumping to an arbitrary node requires an XPath expression. `XmlNode`'s methods `SelectXxx` can be used to find a single node or all nodes that match a given criteria. The method `SelectSingleNode` returns the first node matching the search, from the top of the tree down, in document order. The method `SelectNodes` returns an `XmlNodeList` object, a container for `XmlNode` objects that can be read by a foreach statement:

```
XmlNode root = doc.DocumentElement;
XmlNodeList nodeList =
    root.SelectNodes("..."); // XPath
    expression foreach (XmlNode node in
        nodeList) {
        ... // work with the single nodes
    }
```

The `XmlNodeList` object contains references to the underlying document's nodes. By modifying the value of a node, that node is also updated in the document and vice versa.

For more sophisticated navigation (e.g., over any XML-enabled data store like a database, or also for XSLT), the .NET namespace `System.Xml.XPath` contains a number of classes and interfaces. Its core is the abstract class `XPathNavigator` defining the common functionality of all navigators. The classes `XmlNode` (because it implements the interface `IXPathNavigable`) and `XmlDocument` export the method `CreateNavigator` for creating an `XPathNavigator` object:

```
XmlDocument doc = new XmlDocument();
doc.Load("data.xml");
XmlElement root = doc.DocumentElement;
// or any other node
XPathNavigator navigator =
    root.CreateNavigator();
Console.WriteLine(" " +
    navigator.GetType());
```

It's interesting that `XPathNavigator` is an abstract class with no public subclass in .NET. `CreateNavigator` still delivers an object; its type is unknown to the user. The last line reveals it: `System.Xml.DocumentXPathNavigator`, a class not published in .NET. It might be an inner type of the class `XmlNode`, or perhaps its publication has been forgotten.

This object supports general navigation methods: selecting nodes, iterating over the selection, copying, moving, removing, and so on. The methods of `XPathNavigator` accept an XPath expression in the form of a string or a precompiled expression object; they are evaluated to identify the matching set of nodes.

The class `XPathDocument` is an optimized version of `XmlDocument` for XSLT processing and XPath queries; it provides a read-only, high-performance cache.

## Writing XML

`XmlWriter` is an abstract class (like `XmlReader`) defining the base functionality for producing an XML document. The concept of .NET's `XmlWriter` is very similar to SAX. `XmlTextWriter` is currently its only implementation of the .NET class library; `XmlNodeWriter` (the counterpiece to `XmlNodeReader`) is hopefully coming in a future release. They work just like the reader versions but in the opposite directions.

`XmlTextWriter` provides the `WriteXxx` methods for writing out typed elements and attributes, where `Xxx = StartDocument`, `EndDocument`, `DocType`, `StartElement`, `EndElement`, `FullEndElement`, `ElementString`, `StartAttribute`, `EndAttribute`, `Attributes`, `AttributeString`, `Comment`, `ProcessingInstruction`, `String`, `Base64`, `BinHex`, `CData`, `CharEntity`, `Chars`, `EntityRef`, `Name`, `NmToken`, `Node`, `Qualified-Name`, `Raw`, `SurrogateCharEntity`, or `Whitespace`. It's possible, for example, to write out the whole current node in an `XmlReader` (with or without attributes) by calling `WriteNode`:

```
static void Serialize(XmlReader
    reader, XmlWriter writer) {
    while (reader.Read())
        writer.WriteNode(reader, true);
    // true = with default attributes
}
```

`XmlTextWriter` supports different output stream types (its constructor takes a string [file or URI], a `Stream`, or a `TextWriter`) and is configurable. Its properties specify whether to provide namespace support (bool property



Namespace), indentation options (int property Indentation, char property IndentChar), quote character for attribute values (char property QuoteChar), lexical representation for typed values, and so on. Listing 2 illustrates how some of these properties can be configured:

## XSL Transformations

The namespace System.Xml.Xsl contains the class XslTransform, which manages XSLT transformations. It uses XPathNavigator during the transformation process. XslTransform reads as input an XML document, an XSLT document, and some optional parameters (of type XsltArgumentList). It can produce any text-based output.

To perform a transformation, an XslTransform object must be loaded with the XSLT document (as a DOM tree): the Load method requires a string (URL) parameter. Next an XPathDocument object (instead of XmlDocument – it offers a better performance for XSLT processing) must be created and initialized. Finally, the Transform method executes the transformation:

```
static void TransformXML(string xml,
// name of file to transform
    string xslt, // file name of
stylesheet or XSLT document
    XmlTextWriter writer) {
    XslTransform transformator = new
XslTransform();
    transformator.Load(xslt); //
stylesheet loaded
    XPathDocument document = new XPath-
Document(xml); // DOM tree created
    transformator.Transform(document,
null, writer, null);
}
```

The last call of Transform could take some additional XSLT parameters (XsltArgumentList and XmlResolver) – here they are null.

## Replacing XSLT Functions by C#

Most transformations can be elegantly solved in XSLT. Nevertheless there are some tasks that need special tricks.

Let's assume an XML document with a root element <matrix> contains a number of (let's say  $n$ ) <row> elements and all of them contains the same number  $n$  of <column> elements with the text 1\_1, 1\_2, ..., 1\_n, 2\_1, 2\_2, ..., 2\_n, ..., n\_1, n\_2, ... n\_n:

1_1	1_2	...	1_n
2_1	2_2	...	2_n
...			
n_1	n_2	...	n_n

The task is to exchange the row and column elements (to mirror the table along its diameter):

1_1	2_1	...	n_1
1_2	2_2	...	n_2
...			
1_n	2_n	...	n_n

This transformation is called transposing a matrix. Because XSLT is a functional language with no variables, this task can be solved only recursively (see Listing 3).

The template recursive is called here inside the template recursive; this recursion implements a for-cycle of a procedural programming language (like C#) that doesn't exist in XSLT (for-each can iterate only over the children of a node, it's not the for-cycle of C# or Java!).

The same algorithm can be expressed in C# as shown in Listing 4 (Listings 4–6 are available at [www.sys-](http://www.sys-con.com/xml/sourcec.cfm)

[con.com/xml/sourcec.cfm](http://www.sys-con.com/xml/sourcec.cfm)).

It's remarkable that this C# program doesn't use variables; the different values (between 1 and  $n$ ) of the (constant) parameter COLUMN have been stored on the stack by the repeated calls of the method Recursive. So recursion has been "misused" for a variable – in XSLT (as in any other functional language) this is the only way to solve the problem. If we use C# variables, recursion is evitable (see Listing 5).

Here the counter variable column takes the values 1 through  $n$  instead of the parameter COLUMN in the XSLT program.

The opulence of the .NET library and the object-oriented nature of C# allows another, even simpler solution of hanging over the InnerText objects in the references of the original XmlDocument (i.e., offline) (see Listing 6).

...

As you can see, sometimes a self-programmed XSLT processor can solve a problem in a much simpler manner than a prefabricated one. ☺

ANDREAS@SOLYMOSEI.COM

### LISTING 1

```
static void ProcessMyDocument(XmlReader
reader) {
    while (reader.Read()) {
        switch (reader.NodeType) {
            case XmlNodeType.Element:
                Console.WriteLine("<" + reader.Name);
                while (reader.MoveToNextAttribute())
                    Console.WriteLine("'" + reader.Value +
                    "'");
                Console.WriteLine(">");
                break;
            case XmlNodeType.ProcessingInstruction:
                Console.WriteLine("<?" +
                reader.Name + " " + reader.Value +
                ">"); break;
            case XmlNodeType.Comment:
                Console.WriteLine("<!--" +
                reader.Value + "-->"); break;
            case XmlNodeType.Document:
                Console.WriteLine("<?xml
                version='1.0'?>"); break;
            case XmlNodeType.EndElement:
                Console.WriteLine("</" + reader.Name +
                ">"); break;
            default: // Whitespace, Significant
                Whitespace, Text, CDATA:
                Console.WriteLine(reader.Value); break;
        }
    }
}
```

### LISTING 2

```
XmlTextWriter writer = new
XmlTextWriter(Console.Out);
writer.Namespaces = true; // write
namespace names
writer.Formatting = Formatting.Indented;
// pretty-print
writer.Indentation = 4;
writer.QuoteChar = '\\'; // use ' for
attribute values
writer.WriteStartDocument();
```

```
writer.WriteComment("example for XML
Journal");
writer.WriteProcessingInstruction
("xml-stylesheet",
"type='text/xsl' href=data.xsl");
writer.WriteStartElement("family");
writer.WriteString("Solymosi");
writer.WriteEndElement();
writer.WriteEndDocument();
This code segment produces following
simple XML document:
<?xml version='1.0' encoding='IBM437'?>
<!--example for XML Journal-->
<?xml-stylesheet type='text/xsl'
href=data.xsl?>
<family>Solymosi</family>
```

### LISTING 3

```
<xsl:template match="matrix">
<matrix>
<xsl:call-template name="recursive">
<xsl:with-param name="COLUMN"
select="1"/>
</xsl:call-template>
</matrix>
</xsl:template>
<xsl:template name="recursive">
<xsl:param name="COLUMN" />
<xsl:if test="row/column[ $COLUMN ]">
<row>
<xsl:for-each select="row">
<column>
<xsl:apply-templates select="column[
$COLUMN ]"/>
</column>
</xsl:for-each>
</row>
<xsl:call-template name="recursive">
<xsl:with-param name="COLUMN"
select="$COLUMN + 1"/>
</xsl:call-template>
</xsl:if>
</xsl:template>
```

▼ Download the Code  
▼ [www.sys-con.com/xml](http://www.sys-con.com/xml)



# Ontology and Integration

## Managing Application Semantics Using Ontologies and Supporting W3C Standards

Many in the world of application integration have begun to adopt the notion of ontology (or the instances of ontology: ontologies). Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related.

Ontologies are important to application integration solutions because they provide a shared and common understanding of data (and, in some cases, services and processes) that exists within an application integration problem domain, and how to facilitate communication between people and information systems. By leveraging this concept we can organize and share enterprise information, as well as manage content and knowledge, which allows better interoperability and integration of inter- and intra-company information systems. We can also layer common ontologies within verticals, or domains with repeatable patterns.

When dealing with application integration, as you know by now, we are dealing with much complexity. The notion of ontologies helps the application integration architect prepare generalizations that make the problem domain more understandable. In contrast to abstraction, generalization ignores many of the details and ends up with general ideas. Therefore, when generalizing, we start with a collection of types and analyze commonalities to generalize them.

Clearly, semantic heterogeneity and divergence hinders the notion of generalization, and as commonalities of two entities are represented in semantically different ways, the differences are more

difficult to see. Thus, ontological analysis clears the ground for generalization, making the properties of the entities much more clear. Indeed, ontological analysis for application integration encourages generalization.

Considering that statement, it's also clear that application independence of ontological models makes these applications candidates for reference models. We do this by stripping the applications of the semantic divergences that were introduced to satisfy their requirements, thus creating a common application integration foundation for use as the basis for an application integration project.

One of the benefits of leveraging ontologies is the fact that no matter where the information resides, we can understand and map information relevant to the application integration scenarios. Ontologies allow you to differentiate between resources, which is especially useful when those resources have redundant data (e.g., customer information in almost all enterprises). Thus, in order to make better sense of the data and represent the data in a meaningful way, terms defined in ontologies allow the application integration architects to fully understand the meaning and context of the information. Again, this is ontology's value within application integration.

When considering schemas, local to remote source or target systems, the application of ontologies is leveraged in order to define the meaning of the terms used in some domain. Although there is often some communication between a data model and the attributes, both schema and ontologies play key roles in application integration because of the importance of both semantics and data structures.

Another important notion of ontologies is entity correspondence. Ontologies that are leveraged in more of a B2B environment must leverage data that is scattered across very different information systems, and information that resides in many separate domains. Ontologies in this scenario provide a great deal of value because we can join information together, such as product information mapped to on-time delivery history mapped to customer complaints and compliments. This establishes entity correspondence.

To gather information specific to an entity, we need to leverage different resources to identify individual entities, which vary widely from each physical information store. For example, when leveraging a relational database, entities are identified using keys (e.g., customer number). Within the various information systems, many different terms are used for attributes. The notion of ontologies, in this scenario, allows us to determine whether entities from different applications and databases are the same or noncrucial to fusing information.

### W3C Standards and Ontologies

Resource Description Framework (RDF), a part of the XML story, provides interoperability between applications that exchange information. RDF is another Web standard that's finding use everywhere, including application integration. RDF was developed by the W3C to provide a foundation of metadata interoperability across different resource description communities and is the basis for the W3C movement to ontologies such as the use of Web Ontology Language (OWL).

#### AUTHOR BIO

David S. Linthicum is the former CTO of Mercator, and author of the ground-breaking book *Enterprise Application Integration*. His latest book is *Next Generation Application Integration*.

RDF uses XML to define a foundation for processing metadata and to provide a standard metadata infrastructure for both the Web and the enterprise. The difference between the two is that XML is used to transport data using a common format, while RDF is layered on top of XML defining a broad category of data. When the XML data is declared to be of the RDF format, applications are then able to understand the data without understanding who sent it.

RDF extends the XML model and syntax to be specified for describing either resources or a collection of information. (XML points to a resource in order to scope and uniquely identify a set of properties known as the schema.)

RDF metadata can be applied to many areas, including application integration. One example would be searching for data, and cataloging data and relationships. RDF is also able to support new technology (such as intelligent software agents and exchange of content rating).

RDF itself does not offer predefined vocabularies for authoring metadata. However, the W3C does expect standard vocabularies to emerge once the infrastructure for metadata interoperability is in place. Anyone, or any industry, can design and implement a new vocabulary. The only requirement is that all resources be included in the metadata instances using the new vocabulary.

RDF benefits application integration in that it supports the concept of a common metadata layer that is sharable throughout an enterprise or between enterprises. Thus, RDF can be used as a common mechanism for describing data within the application integration problem domain.

The use of languages for ontology is beginning to appear, built on reasoning techniques that provide for the development of special-purpose reasoning services. In fact, the W3C has created a Web standard for ontology language as part of its effort to define semantic standards for the Web. The Semantic Web is the abstract representation of data on the World Wide Web, based on the Resource Description Framework standards and other standards still to be defined. It is being developed by the W3C, in collaboration with a large number of researchers and industrial partners led by Tim Berners-Lee.


In order for the Semantic Web to function, computers must have access to structured collections of information

and sets of inference rules that they can use to conduct automated reasoning. This notion is known as knowledge representation. To this end, and in the domain of the World Wide Web, computers will find the meaning of semantic data by following hyperlinks to definitions of key terms and rules for logically reasoning about data. The resulting infrastructure will spur the development of automated Web services such as highly functional agents. What's important here is that the work now being driven by the W3C as a way to manage semantics on the Web is applicable, at least at the component level, to the world of application integration, much like XML and Web services.

An example of the W3C contribution to the use of ontologies is the Web Ontology Language. OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is derived from the DAML+OIL Web Ontology Language and builds upon the RDF. OWL assigns a specific meaning to certain RDF triples. The future Formal Specification at the W3C specifies exactly which triples are assigned a specific meaning, and offers a definition of the meaning. OWL provides a semantic interpretation only for those parts of an RDF graph that instantiate the schema. Any

additional RDF statements resulting in additional RDF triples are allowed, but OWL is silent on the semantic consequences of such additional triples. An OWL ontology is made up of several components, some of which are optional, and some of which may be repeated.

Using these Web-based standards as the jumping-off point for ontology and application integration, it's possible to define and automate the use of ontologies in both intra- and intercompany application integration domains. Domains made up of thousands of systems, all with their own semantic meanings, are bound together in a common ontology that makes short work of application integration and defines a common semantic meaning of data.

This, indeed, is the goal. Extending from the languages, we have several libraries available for a variety of vertical domains, including financial services and e-business. We also have many knowledge editors that now exist to support the creation of ontologies, as well as the use of natural-language processing methodologies. We have seen these in commercially available knowledge mapping and visualization tools using standard notations such as UML. 

LINTHICUM@ATT.NET

XML-J ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com	978-816-1600	52
Active Endpoints, Inc.	www.active-endpoints.com	203-929-9400	11
Assande	www.assande.com		2
CTIA	www.ctiashow.com		17
Edge 2004 East	www.sys-con.com/edge	201-802-3069	25-40
Ektron	www.ektron.com/xmlj	603-594-0249	51
Linux World Magazine	www.linuxworld.com	888-303-5282	41
Mindreef	www.mindreef.com	603-465-2204	4
Web Services on Wall Street			19

**General Conditions:** The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *XML-Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *XML-Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

WRITTEN BY STANKO BLATNIK,  
ROUSLAN KADYROV & KELLY CAREY

# Monitoring Air Pollution in Real Time Using XML

## A reliable and cost-effective open source solution

### AUTHOR BIO

Stanko Blatnik teaches Web development (XML, SVG) online for the University of Sarajevo and directs a non-profit institute in Velenje, Slovenia: the Institute for Symbolic Analysis of Information Technologies. Stanko has a doctorate degree in theoretical particle physics from the University of Zagreb.

Ruslan Kadyrov has a doctorate degree in solid state physics from University at Tomsk, Siberia. Ruslan's work includes researcher in Material Science Institute at Tomsk and software developer at Jozef Stefan Institute at Ljubljana, Slovenia. Currently, he works as leading programmer at Artes company in Velenje, developing complex applications using Java and XML.

Kelly Carey teaches digital media/Internet services courses at West Valley College in Saratoga, California. Kelly has a doctorate degree from the University of San Francisco in organizational leadership with an emphasis on the Pacific Rim.

A few years ago there was no indication that XML could play an important role in computer-based process control. However, fast development and the spread of XML to different fields, and emerging trends in the field of automation, have changed the situation tremendously.

Several factors support the application of XML in this field:

- The penetration of the Internet and intranets in manufacturing processes
- The need to use a standard, nonproprietary format for data exchange
- Interconnection between business information and process control systems

This article describes an XML-based system used for monitoring air quality at Tuzla Canton, Bosnia-Herzegovina. The system includes measurement stations at different locations, as shown in Figure 1.

After user analysis and discussions with the client, the Cantonal Environmental Protection Agency at Tuzla, the decision was made not to use the routine solution to the problem, the SCADA (supervisory control and data acquisition) system. The main arguments for this decision were that the SCADA system has a multi-dimension structure, and several features would be not used in this application. In addition, the price for SCADA would be significantly higher and developing the SCADA application would take longer than developing a solution using open source resources. Another advantage of this approach is that it's easier to make custom-designed applications than to work with SCADA, which isn't so flexible.

Several measurement stations collect air quality data, which must be processed and stored in local systems and sent to the central computer where

it is processed and stored in a database. If alarm messages are sent, the corresponding activity is performed. The data stored in the archive can be analyzed and shown to users – clients on the intranet who can demand different reports that are available for different times from different locations.

In this system there are several points where XML technologies can be used to improve efficiency. Here are some possibilities:

- Creation of an XML document at the local station that will be sent to the central computer
- Analysis of an arriving XML document at the central computer searching for alerts
- Storage of the data from an arriving XML document to the database in the central computer
- Extraction of data from the archive and generation of an XML document that will be processed on the server and sent to the user
- Use of XSLT for creation of an SVG document that will be transferred using Batik to graphic presentation on the user's PC

### The Solution

After discussion with the client, the Cantonal Environment Protection Agency, the following solution was developed.

Data from each station's measurement devices is sent to an OPTO 22 PLC (programmable logical controller) system. The OPTO 22 generates XML text string messages. This is an advantage compared to the other PLC systems, which generate messages as streams of bits. XML documents travel over phone lines to a central computer. An application program uses XSLT to generate new XML documents in a more suitable form. The central computer parses these XML

documents for data analysis using JDOM.

JDOM is, quite simply, a tool that creates a Java representation of an XML document. JDOM provides a way to represent that document for easy and efficient reading, manipulation, and writing. It has a straightforward API, is lightweight and fast, and is optimized for the Java programmer. It's an alternative to DOM and SAX, although it integrates well with both.

JDOM generates and sends new XML documents to the database where the data is stored and to the Java Message Service (JMS). The JMS API improves programmer productivity by defining a common set of messaging concepts and programming strategies that will be supported by all JMS technology-compliant messaging systems where the type publish is used.

The central computer, JBoss server (intranet), connects clients. Users on the client select different applications generated from the XML document sent from the server. The screen generation process consists of the following steps:

1. Transformation of XML to the SVG document using XSLT
2. The Java application uses the Batik library for presenting SVG on the user's screen

The following data is collected from the measurement stations.

- Concentration of sulphur oxides (SOx)
- Concentration of nitrogen oxides (NOx)
- Concentration of dust particles
- Temperature
- Direction of wind
- Velocity of wind
- Air temperature
- Air humidity
- Precipitation
- Air pressure
- Sun radiation



- Concentration of carbon monoxide (CO)
- Concentration of ozone (O3)
- Concentration of methane

Data is collected from different instruments connected to the OPTO 22 PLC system, organized in the XML document (part of it is shown in Listing 1; code listings are available at [www.sys-con.com/sourcec.cfm](http://www.sys-con.com/sourcec.cfm)), and sent to the central computer as text strings. Here the elements present the following data:

- <reply> is the root element whose attributes contain information about station type, its identification number, and the time when the reply was sent to the central computer.
- <parameter> element with the attribute value of "name equals system" contains data about system settings.
- <parameter name="WIND"> contains information about wind.
- <parameter name="SO2"> contains information about sulphur dioxide concentration.
- <parameter name="NO"> contains information about nitrogen monoxide concentration.

The other measurements are described in same way.

The XML document that presents the status of devices at different measurement stations is transformed by the XSLT document for further processing; part of it is shown in Listing 2.

This XSLT document generates the XML shown in Listing 3, which is used in further processing. This transformation is realized from a Java application and JDOM; a part of it is shown in Listing 4.

The new XML document is sent to the Web server using JMS; the Web server is realized under JBoss. In the application layer, the XML document is parsed and analyzed. Depending on the incoming data, different actions can be triggered. For example, if the SO2 concentration is higher than acceptable, an alarm is generated.

On the application level, JDOM navigates and modifies the XML document. Data extracted from the XML document is saved in MySQL using JDBC. Data, rather than the entire XML document, is saved so it can be analyzed later in the context of a different application.

JDOM generates new XML that is sent to other clients, as shown in Listing 5.

Users on clients select different applications by clicking on different icons on the map. Applications are generated using XSLT stylesheets to transform XML documents sent by the Web server, in the SVG document. Listing 6

shows part of a typical XSLT stylesheet that dynamically generates corresponding SVG documents.

The user interface is Java, and the Batik library supports drawing the SVG documents onscreen. The following classes are imported from the Batik, DOM, and SVG libraries.

```
import org.apache.batik.swing.JSVGCanvas;
import org.apache.batik.swing.gvt.
    GVTTreeRendererAdapter;
import org.apache.batik.swing.gvt.
    GVTTreeRendererEvent;
import org.apache.log4j.Logger;
import org.w3c.dom.Document;
import org.w3c.dom.svg.SVGDocument;
```

Running Java code generates Figures 2 and 3.

## Conclusion

Typically, applications such as the one described in this article are realized using a SCADA system. SCADA systems are complex and offer several options that have not been used in this application. The client asked for cheap and reliable solutions, so we decided to use XML in combination with JBoss and MySQL. Because we used JDOM and a library of Java classes, the development time was relatively short, and minimal programmer resources were required.

Our approach has several advantages:

- It is based on open source and is platform independent. When developed, it operates on all systems because it uses Java and XML technologies. Most SCADA systems work under Windows operating systems and can present significant costs for the whole system.
- It is more flexible in that tasks are easily distributed between programmers.
- To add a new application, only a new XSLT file needs to be developed.
- The programming package can easily be extended to adapt to new requirements.

The system was successfully installed and tested, and works effectively as a real-time application in the Canton of Tuzla in Bosnia-Herzegovina. The system was developed and implemented by Artes, a small Slovenian company specializing in IT applications in power plants and environmental pollution monitoring using XML and Java.

We think that the approach described in this article will be used in the future in systems that have a large collection of data that has to be processed and shown on an intranet. One example of a potential application is the development of an intranet-based



Figure 1 • Measurement station locations



Figure 2 • Emissions and meteorological data from a specific location

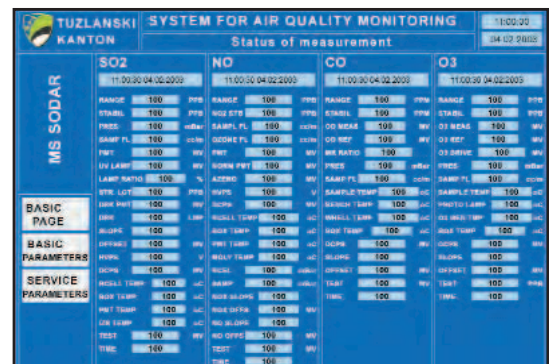


Figure 3 • Status of measurement from a specific location

information system for a distributed electricity production system. ☼

## Resources

- Kelly Carey and Stanko Blatnik. *XML Content and Data*. Prentice Hall, 2002.
- Kelly Carey and Stanko Blatnik. *Design Concepts with Code: An Approach for Developers*. Apress, 2003.
- JBoss: [www.jboss.org/index.html](http://www.jboss.org/index.html)
- JDOM: [www.jdom.org](http://www.jdom.org)

...

Acknowledgements: The authors would like to thank Artes Group, Jure Londrandt, Damijan Onisak and Slavko Sadl.

KELLY-CAREY@YAHOO.COM  
STANKO-BLATNIK@YAHOO.COM



## LISTING 1

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE reply SYSTEM "../DTD/request-reply.dtd">
<reply station="MOBILE" id="12" time="10:03:20 09/01/2003">
  <status time="10:03:20 09/01/2003" samples="0">
    <parameter name="SYSTEM" >
      <function type="DI" name="SETTINGS" value="262049"
        unit=" " />
    </parameter>
    <parameter name="WIND" >
      <function type="DI" name="VECT_SPEED" value="0.00"
        unit="m/s" />
    </parameter>
    <parameter name="SO2" >
      <function type="DI" name="CONCEN" value="0.00"
        unit="ug/m3" />
    </parameter>
    <parameter name="NO" >
      <function type="DI" name="CONCEN" value="0.00"
        unit="ug/m3" />
    </parameter>
  </status>
</reply>
```

## LISTING 2

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:java="http://xml.apache.org/xslt/java">
<xsl:output method="xml" version="1.0" indent="yes"
  encoding="iso-8859-2"
  cdata-section-elements="script"/>
<xsl:template match="status">
  <xsl:element name="taglist">
    <xsl:apply-templates select="parameter/function"/>
    <xsl:apply-templates select="service"/>
  </xsl:element>
</xsl:template>
<xsl:template match="parameter/function">
  <xsl:element name="tag">
    <xsl:attribute name="name"><xsl:call-template name=
      "stationToUpperCase"/><xsl:value-of
        select="../@name"/><xsl:value-of
        select="@name"/></xsl:attribute>

    <xsl:attribute name="timestamp"><xsl:call-template
      name="timeFormat"/></xsl:attribute>
    <xsl:value-of select="@value"/>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

## LISTING 3

```
<?xml version="1.0" encoding="iso-8859-2"?>
<taglist>
  <tag name="MOBILE:SYSTEM.SETTINGS_STATION">
    <source-address>MOBILE:SYSTEM.SETTINGS_STATION</
      source-address>
    <value-type>java.lang.Integer</value-type>
    <values>
      <value timestamp="09.01.2003 10:03:20">269</value>
    </values>
  </tag>
  <tag name="MOBILE:WIND.SPEED">
    <source-address>MOBILE:WIND.SPEED</source-address>
    <value-type>java.lang.Double</value-type>
    <values>
      <value timestamp="09.01.2003 10:03:20">0.00</value>
    </values>
  </tag>
  <tag name="MOBILE:WIND.DIRECTION">
    <source-address>MOBILE:WIND.DIRECTION</source-address>
    <value-type>java.lang.Double</value-type>
```

```
<values>
  <value timestamp="09.01.2003 10:03:20">0.0</value>
</values>
</tag>
<tag name="MOBILE:SULPHUR.CONCENTRATION_SO2">
  <source-address>MOBILE:SULPHUR.CONCENTRATION
    _SO2</source-address>
  <value-type>java.lang.Double</value-type>
  <values>
    <value timestamp="09.01.2003 10:03:20">0.00</value>
  </values>
</tag>
</taglist>
```

## LISTING 4

```
public Document transform(Document documentJDOMEntree,
  String xsltFile) throws Exception
{
  TransformerFactory factory = TransformerFactory.
    newInstance();
  transformer = factory.newTransformer(new StreamSource
    (xsltFile));
  JDOMResult res = new JDOMResult();
  transformer.transform(new JDOMSource(documentJDOMEntree),
    res);
  return res.getDocument();
}
```

## LISTING 5

```
<?xml version="1.0" encoding="iso-8859-2"?>
<display-data name="MAP">
  <tag key="so2_MP1" name="SIMULATE::OCYH01CQ001ZM01"/>
  <tag key="dust_MP1" name="SIMULATE::OCYH01CQ008ZM01"/>
  <tag key="wind_MP1" name="SIMULATE::OCYH01CR001ZM01"/>
  <tag key="door_MP1_DI" name="SIMULATE::OCYH01EG001ZS00"/>
  <tag key="time_system" name="SIMULATE::SYSTEMTIME"/>
  <tag key="time_MP1" name="SIMULATE::MP1TIME"/>
  <tag key="station_name1" name="MS1 - Square"/>
</display-data>
```

## LISTING 6

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<xsl:output method="xml" version="1.0" indent="yes"
  encoding="windows-1250" cdata-section-elements="script"/>
<xsl:strip-space elements="*" />
<xsl:template match="display-data">
  <!--The measured values which will be displayed-->
  <xsl:variable name="SO2.MP1"><xsl:value-of
    select="tag[@key='so2_MP1']/values/value"/></xsl:variable>
  <xsl:variable name="DUST.MP1"><xsl:value-of select=
    "tag[@key='dust_MP1']/values/value"/></xsl:variable>
  <xsl:variable name="WIND.MP1"><xsl:value-of select=
    "tag[@key='wind_MP1']/values/value"/></xsl:variable>
  <xsl:variable name="DOOR.MP1"><xsl:value-of select=
    "tag[@key='door_MP1_DI']/values/value"/></xsl:variable>
  <xsl:variable name="STATIONNAME1"><xsl:value-of select=
    "tag[@key='station_name1']/values/value"/></xsl:
    variable><xsl:variable name="angle1"><xsl:if
      test="$WIND.MP1='????'">0</xsl:if><xsl:if
      test="$WIND.MP1!='????'"><xsl:value-of
        select="($WIND.MP1-90)/></xsl:if></xsl:variable>
  <xsl:variable name="DATE.SYSTEM"><xsl:value-of select=
    "substring(tag[@key='time_system']/values/value,1,10)"
    /></xsl:variable>
  <xsl:variable name="WATCH.SYSTEM"><xsl:value-of select=
    "substring(tag[@key='time_system']/values/value,12,8)"
    /></xsl:variable>
  <xsl:variable name="TIME.MP1"><xsl:value-of select=
    "tag[@key='time_MP1']/values/value"/></xsl:variable>
  <svg viewBox="0 0 800 600" width="100%" height="100%"
    onload="blinking(event)">
  <!--Here is SVG with maps and symbols to be displayed-->
  </svg>
</xsl:template>
</xsl:stylesheet>
```

## Ektron Success Story #1531



A leading baking products company needed to build a website that reflected brand and offered consumers access to up-to-date product information, recipes and promotional programs. With Ektron's Web content management solution they were able to achieve a high degree of site functionality and automation including the use of XML and Web Services to syndicate content to internal and third party web sites.

**Enterprise Web Content Management without the enterprise integration.**



**Let us show you how an Ektron XML Web Content Management Solution can enhance your Web site.**

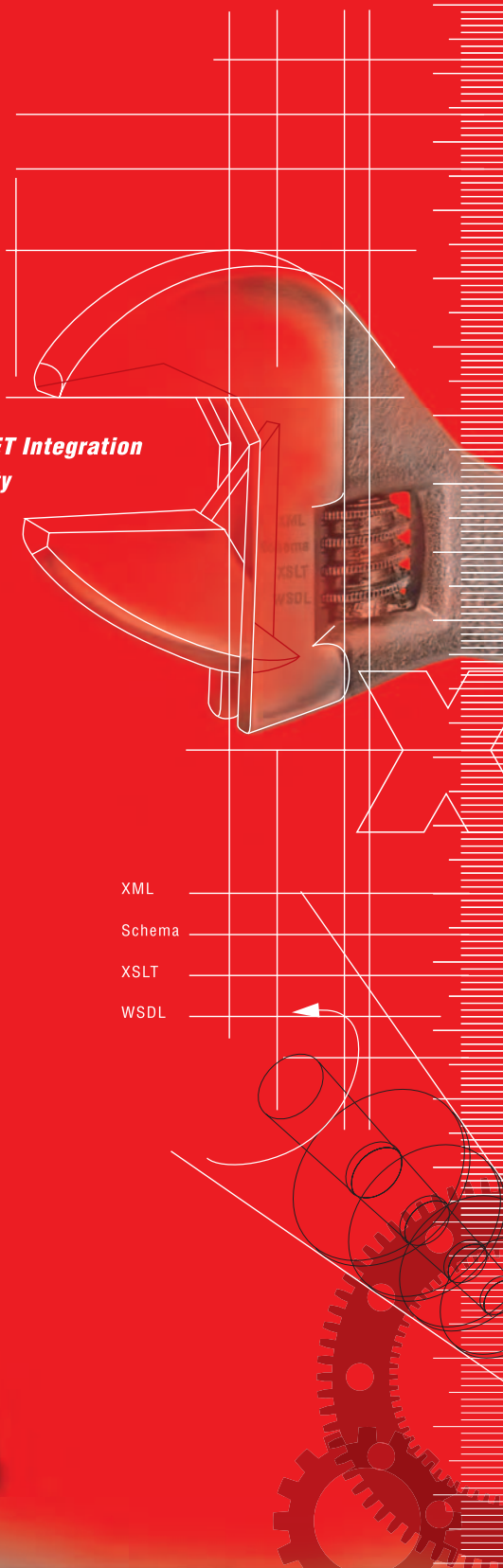
**Learn More at:**  
[www.ektron.com/xmlj](http://www.ektron.com/xmlj)

**Ektron - Redefining Web Content Management**



**NEW** Microsoft® Visual Studio®.NET Integration  
Enhanced Database Connectivity  
XML Differencing  
XPath 2.0 Analyzer

improved XML Schema/WSDL Editor  
XSLT Debugger and Code Generator



### Introducing mapforce™ 2004

New Visual Database-to-XML and XML-to-XML mapper!  
Generate Java, C#, C++, and XSLT



Accelerate your next XML development project with the leading XML tools from Altova! **xmlspy® 2004** is the newest release of the industry standard XML Development Environment for modeling, editing and debugging any XML technology. **mapforce™ 2004** is an all-new, visual data mapping and code generation tool. Specially priced product bundles are available, see website for details.

Download xmlspy® 2004 and mapforce™ 2004 today: [www.altova.com](http://www.altova.com)

**ALTOVA®**

[www.altova.com](http://www.altova.com)