

XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

February 2004 Volume: 5 Issue:2

XML-JOURNAL.COM

FROM THE EDITOR

Raising the Bar

Open standards are good –
now let's prove it with ROI
by Hitesh Seth **pg. 3**

COMMENTARY

XML and Web Services: Connecting Information Islands

How savvy content
owners deliver relevant
search results
by Kevin Migliozi **pg. 7**

STANDARDS

Remember ebXML?

Doing business in real time
by David S. Linthicum **pg. 30**



An architectural overview of WebSphere Business Integration Connect **6**

Feature: Community Integration & XML

Scott Simmons

Extending integration between trading communities

Community	Trading Partner	Integration	Integration
IBM	IBM	IBM	IBM
IBM	IBM	IBM	IBM
IBM	IBM	IBM	IBM
IBM	IBM	IBM	IBM
IBM	IBM	IBM	IBM

6

First-Class Services for the Travel Industry

Anjan Mitra

Managing the quality of Web services systems



12

Troubleshooting .NET Applications

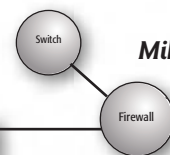
Karthik Ravindran

Implementing an XslTransform wrapper to trace XSLT transformations

14

Matters of Syntax

ConciseXML builds upon
the important qualities of XML and S-Expressions



Mike Plusch

20

FileAct for SWIFTNet

Helping financial services gain savings from their network

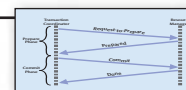


Cathy Kingeter

24

Web-Services Transactions

Loosely Coupled–The Missing Pieces of Web Services



Doug Kaye

26

Rendering a Connected Tree Using XSLT

The ideas behind XSLT and XPath make perfect sense



Craig Berry

32

edge 2004 EAST
DEVELOPMENT TECHNOLOGIES EXCHANGE
February 24-26, 2004
Hynes Convention Center, Boston, MA
ARCHITECTING JAVA, .NET, WEB SERVICES, MX, XML, AND OPEN SOURCE

FIVE FLAVORS FOR IT PROFESSIONALS

DISPLAY UNTIL April 30, 2004

\$6.99US \$7.99CAN



0 09281 01314 3

BEA WebLogic Workshop 8.1 makes you more

AMAZING.



Development Tools
December 30, 2003
BEA WebLogic Workshop 8.1
BEA Systems, Inc.

"A shared, consistent
development platform...for
J2EE experts and business-
oriented developers alike."



Java Pro Readers
Choice Award

"WebLogic Workshop
empowers all application
developers...not just J2EE
developers."



Software Development Jolt Pro-
duct Excellence Award

"The integration of an IDE,
controls and a deployment
environment...greatly enhances
developer productivity."



CRN Test Center
Product of the Year

"By far the most innovative
development tool reviewed
this year...when compared
with other tools, Workshop
blew away the competition."

But don't take their word for it, find out for yourself now, for free.

BEA's WebLogic Workshop 8.1 has won every major award for software development tools in the past year. It's a full-featured Java development environment that lets you visually build and assemble enterprise-scale Web Services, Web Applications, JSPs, Portals, EJBs, and Business Process models based on the latest standards and open source technologies.

Get it now. For free. Visit <http://dev2dev.bea.com/workshop3>

dev2dev™



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

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Craig Berry, Doug Kaye, Cathy Kingeter,
David S. Linthicum, Kevin Migliozi, Anjan Mitra,
Mike Plusch, Hitesh Seth, Scott Simmons

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.



Raising the Bar

WRITTEN BY HITESH SETH



2003 was definitely a positive year for the economy and the world of i-technology – and the year's success also raised the bar for what needs to happen in 2004, if we really want to see recovery. Although this statement focuses on financial aspects, it provides an indication of what we need in the world of information technology – a focus on delivering value out of the whole standardization story.

We've continuously said that using open standards such as XML and Web services is a good thing – we need to start proving it through implementations whose success can be attributed to these standards. As the economy improves, spending should also increase and more investment dollars will be available. However, instead of spending these dollars on long-term research projects, the business side of the world wants to see projects that bring immediate ROI.

Standards

The success of XML has been attributed to the development of standards. Whereas the W3C has taken on the responsibility of the core set of standards around XML and Web services, OASIS has been instrumental in developing standards that are high up in the technology stack. In addition to W3C and OASIS, WS-I (Web Services Interoperability Organization), which was launched in 2003, has been specifically targeted to focus on interoperability across platforms and applications.

W3C

As you're probably aware, the standardization work at W3C follows a known path. W3C Working Groups take the development of a standard through a series of stages: Working Draft, Last Call Working Draft, Candidate Recommendation, Proposed Recommendation, and W3C Recommendation. Related to XML activities, 2003 marked the release of SOAP 1.2, XForms, XML Events, SVG 1.1, XPointer, and PNG as W3C Recommendations. Key standards to watch in 2004, which are in later stages of maturity, include VoiceXML 2.0 and OWL. Although XForms was released as a W3C Recommendation, we haven't really seen a large number of implementations nor the ubiquity I believe the standard deserves. Hopefully, in 2004 we'll take this standard to the next level.

OASIS

Key standards earning the "OASIS Standard" stamp include XACML (Extensible Access Control Markup Language), UDDI v2, WSRP (Web Services for Remote Portals), XCBF (XML Common Biometric Format), SAML (Security Assertion Markup Language) 1.1, and SPML (Service Provisioning Markup Language). A number of additional OASIS Technical Committees (TCs) were formed, particularly Business Process Execution Language, Composite Application Framework for Web Services, and Web Services Management.

WS-I

The WS-I Organization delivered WS-I Basic Profile and even Sample Applications for Basic Profile. This led to core interoperability between Web services developed using Microsoft .NET and J2EE platforms. J2EE 1.4 included support for WS-I as a highlight of the release. Key plans for 2004 are centered on interoperability of Web services security.

Other Key Highlights

We shouldn't forget that the standardization committees and working groups are powered by member technology and end-user companies, and provide key insight around the development of technology.

In addition to standardization, 2003 also saw success in various other areas. The rubber really met the road for Web services, with initial success from Amazon and Google Web services. A number of Web sites and application models emerged based on those two Web services application areas. This really proved the whole Web services theory and also provided direction for getting value from open standards.

The release of Office 2003 was also a landmark in the history of technology – interoperable office documents are a good thing. Even though there is contention on a broader XML format for Office, with OASIS leading an initiative around it, there's still a lot of value in having some standard – whether it's a recognized standard or a de facto standard. In addition to XML for Word and PowerPoint documents, the Office 2003 release contains a revolutionary killer app that I believe will really change the way we have come to use electronic forms and applications (more next month).

AUTHOR BIO

Hitesh Seth is editor-in-chief of XML-Journal and writes regularly on XML.

Web services, J2EE, Microsoft .NET, and integration technologies.



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>

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 Managers

Alisa Catalano alisa@sys-con.com

Carrie Gebert carrieg@sys-con.com

Associate Sales Managers

Kristin Kuhnle kristin@sys-con.com

Beth Jones beth@sys-con.com

SYS-CON EVENTS

President

Grisha Davida grisha@sys-con.com

Conference Manager

Lin Goetz lin@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

Linda Lipton linda@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

XML and Web Services: Connecting Information Islands

WRITTEN BY KEVIN MIGLIOZZI



The Web has become the world's greatest repository of information on anything and everything. It's extremely useful – as long as you can find what you're looking for.

Despite the arrival of Google and other powerful search tools, information seekers can't always connect with the most relevant information. Searches often are too broad and yield irrelevant results. But there's an even more problematic issue: digital information often remains undiscovered – stranded on "islands," out of reach of Web site or intranet users.

Increasingly, however, savvy content owners are turning to XML and Web services to deliver relevant search results to seekers of Web information in a standard way. Through the use of XML authoring tools, owners of these digital information islands are simplifying the creation of XML content and enabling them to more effectively connect with consumers of information.

Business are realizing the many benefits of connecting these information islands: making information more accessible and thus more valuable; providing more relevant results to Web site visitors; and improving the efficiency and effectiveness of their organizations.

XML defines – or "tags" – Web content in a way that makes it easy for information to be exchanged or located. Combined with Web services, it serves as unifying force that makes content more accessible and available. When information is more accessible, it's open to a larger audience, making it exponentially more valuable to your organization.

Ektron expects 2004 to be a landmark year for the adoption of XML content made available via Web services. More and more companies will design their Web strategies using this technology. XML and Web services will enhance the search, retrieval, exchange, storage, editing, publishing, management, use, and reuse of information.

For example, XML can help real estate agencies succeed in connecting homebuyers and sellers by connecting islands of information without a need to duplicate data – generating more opportunities for sales. Consider the scenario of a real estate agent in California whose Web site serves up a localized database of homes for sale. The site needs to connect to databases in other realtors' offices across the country to serve home listings in other markets.

In one instance, a buyer in California visiting their local realtor's site may be looking for the following criteria in a home: "New Hampshire" and "three bedroom" and "minimum

1,400 square feet". The local California realtor doesn't have that information in their database. You can't search successfully on Google based on that criteria. But thanks to the power of XML-tagged content and Web services, that California real estate Web site can call up information from other realtors' Web servers and serve it up to users via a browser.

The use of XML content also has a significant relevance to corporate intranets. Take the example of a Fortune 500 corporation with thousands of employees in multiple facilities. When an employee in one facility searches the local intranet for information, Web services may call out a query to other intranet servers across the company. It can search the XML-tagged information on those disconnected islands and produce highly relevant search results, benefiting the employee (who finds the information they need) and the organization (which delivered quick and efficient service through its intranet).

Content management vendors continue to empower the XML revolution by developing tools that put the power of XML into the hands of any content owner. For example, content management solutions (CMS) provide fast retrieval of content within any element of XML documents. This enables advanced search operators and sophisticated indexing technology – users benefit from virtually unlimited searching possibilities. It gives users the ability to search metadata and the full text of documents, allowing them to query the server using multiple indexes simultaneously. It also enables searches on full-text, strings, numeric, and date ranges. Extensive multicriteria sorting on any element or document attribute allows maximum flexibility in ordering presented results.

Editing tools continue to make XML creation transparent to content owners. Organizations are creating XML-tagged content for product catalogs, publication listings, contact databases and much more. Organizations also enjoy the benefit of being able to repurpose XML content for various client devices – Web browsers, printers, PDAs, and Web-enabled phones.

As we head into 2004, there will be a strong momentum for organizations to bring Web services technologies into the mainstream of their service-oriented architecture. ☼

AUTHOR BIO

Kevin Migliozi, chief technology officer of Ektron, brings more than 15 years of technical and managerial experience to the company. Kevin's primary role includes investigating emerging technologies and their utilization.

KEVIN.MIGLIOZZI@EKTRON.COM



COMMUNITY INTEGRATION AND XML

WRITTEN BY
SCOTT SIMMONS

Extending integration between trading communities

Community integration elevates collaborative commerce to a new level of integration between enterprises. XML-based document exchanges between companies impose new challenges on organizations building a B2B community integration solution.

In the past, traditional interenterprise solutions have fallen short of market expectations for community integration due to scalability issues, lack of document visibility across the trading community, and minimal partner participation in community management. To solve the tough challenges of XML-based document exchange, new integration solutions must address these requirements to provide an optimal foundation for trading partner collaboration.

This article discusses the concept of community integration and offers an architectural overview of how WebSphere Business Integration Connect can enable the rapid implementation of trading communities.

What Is Community Integration?

Community integration is about extending integration

beyond the enterprise. It defines a document exchange framework that optimizes business processes across enterprise boundaries. Within organizations, there are numerous interenterprise relationships with external partners, some manual and others automated. Over the last 15 years, EDI has been established as a proven foundation for automating these relationships. The emergence of XML document standards such as RosettaNet, CIDX, and others has enabled the evolution of an open communication framework for interenterprise integration. However, providing an XML document exchange for these relationships does not provide community integration in itself.

Inherently each partner interaction is specialized in terms of communication transport, document exchange formats, and quality of service requirements. As a result, the framework for B2B integration must be both flexible and scalable and should allow for the shared management of the relationship at the hub and the partner/participant layer. Community integration is realized when the trading hub provides management access and visibility across the community to the hub and participant constituencies.

Traditional B2B solutions have been of two basic varieties: VAN-based offerings built on FTP/File Transfer EDI communication interchanges and point-to-point solutions for Internet-mediated XML-based document exchange. In both cases, the integration is predicated on point-to-point connections – each connection configured, implemented, and managed separately. These solutions provide only a limited ability to enable partners to provide community management and have not provided global visibility into hub operations. Additionally, the effort required at the hub to manage these communities in the face of technology, economic, and business changes becomes nearly impossible. More important, both of these solution architectures promote a ready-fire-aim mentality that results in a reactive approach to building and managing trading partner relationships.

Community integration provides a foundation for scaling trading communities into the hundreds and thousands of participants. By providing filtered visibility for partners into hub operations and an integrated approach to alert/exception management for both the hub and the partner community, this document exchange architecture enables shared operational responsibility across the community. This shared management focus allows the hub to rapidly add processes and partners to the community while at the same time reducing the overall incremental cost by partner to operate the trading community.

Community Integration and WebSphere Business Integration Connect

WebSphere Business Integration Connect is a community integration solution that was launched via the partnership between Viacore and IBM. Viacore is a service provider offering a comprehensive suite of community integration services for private trading communities. The initial component of WebSphere Business Integration Connect embodies WebSphere software and Viacore's Business Tone Services.

As an example, a major U.S. electronics distributor embarked on a major initiative to align a complex supply chain. The organization needed to quickly implement a private trading network to exchange real-time business processes with its suppliers and customers. This initiative required the synchronization of external document exchanges with internal business processes to support integrated order management, forecasting, and supply-chain planning. By utilizing a community integration framework from Viacore the company successfully implemented an integration solution providing collaborative management and transaction visibility across the trading community. The use of Viacore's BusinessTone solution allows the organization to react more quickly to exceptions, shortages, and trading partner issues and, as a result, these factors drove increased customer satisfaction enabling a more proactive real-time organization.

WebSphere Business Integration Connect is a Java-based software component architecture that was developed in three editions to meet the needs of different-sized businesses:

- Express is designed for small to medium-sized businesses that want to integrate small trading communities with an intuitive, Web-based integration solution. The solution leverages AS2 and HTTP standards for transmitting documents securely over the Internet.
- Advanced Edition enables the creation of larger trading communities that require more sophisticated community management tools. It is ideal for trading communities that will start small and grow over time. The Advanced Edition enables integration over numerous transports and offers a rich set of tools for deep community integration.
- Enterprise Edition is the solution for deploying large trading

hubs with support for unlimited connection definitions. The Enterprise Edition shares the same core technology as the Advanced Edition, differing only from a licensing perspective.

Advanced Edition and Enterprise Edition – an Architectural View

WebSphere Business Integration Connect 4.2 is implemented on the embedded version of the WebSphere Application Server (v5.0.2) and inherits all of the functions of the WebSphere Application Server. WebSphere Business Integration Connect provides support for standards-based transports (HTTP, HTTPS, SMTP, FTP, JMS/WebSphere MQ), message protocols (RNIF 1.1, RNIF 2.0, AS1, AS2), and a wide range of payload formats (XML, RosettaNet, cXML, EDI, Binary/Raw Payloads, SOAP/HTTP/HTTPS, and cXML). SOAP integration includes support for both RPC and Document Styles and uses the mediation/proxy support functionality found within the IBM WebServices Gateway to coordinate and manage internal/external service endpoints.

Examining the product from a document perspective (see Figure 1), there are five basic processing layers that control document flow. Inbound documents (external communication) are handled via a network/transport management layer. Following document receipt, additional message format processing is performed, which includes message format-specific processing, packaging/depackaging, and/or content vali-

WebSphere Business Integration Connect 4.2 Platform Support

WebSphere Business Integration Connect Advanced/Enterprise 4.2 currently runs on Intel platforms. The software runs under Red Hat Linux AS 2.1. The WebSphere Business Integration Connect Advanced/Express 4.2.1 release in December will support IBM pSeries (with AIX 5.2) and Sun SPARC (Solaris V8). Additionally, the 4.2.1 release will support SUSE LINUX ES V8 and Windows 2000 on Intel platforms.

For the 4.2 WebSphere Business Integration Connect Advanced/Enterprise release, the recommended Intel architecture is a 2GHz Intel Xeon processor with 300MB available disk space for application and additional disk space as needed for document storage. As discussed in the article, additional servers can be added for capacity and/or redundancy – multi-server installations require implementation of a shared file system to support distributed document access.

Software requirements for WebSphere Business Integration Connect Advanced/Enterprise include:

- Database: DB2 8.1 FP 2 or Oracle Database Server, Version 9.2 with Oracle JDBC thin driver (Oracle support is with the 4.2.1 release)

- WebSphere MQ, v5.3 or later
- Web Browser: Microsoft Internet Explorer, 5.0 or higher; Netscape, 6.0 or higher for Community Console access
- Simple Mail Transport Protocol (SMTP) server for e-mail alert delivery and SMTP-based message delivery

WebSphere Business Integration Connect Express 4.2 requirements are as follows:

- 1.4 GHz or faster Intel Xeon processor
- At least 512MB of Random Access Memory (RAM)
- At least 100MB of available hard disk space
- Microsoft Windows 2000 operating system, with Service Pack 3 installed
- Microsoft Internet Explorer, 5.5 or higher or Netscape, 6.0 or higher for Express Console access
- A Simple Mail Transport Protocol (SMTP)-based e-mail relay server for delivering e-mail alerts

Note: WebSphere Business Integration Connect Express 4.2.1 will also provide support for Red Hat Linux AS V2.1 and SUSE LINUX Enterprise Server V8 with SUSE LINUX, kernel 2.4.

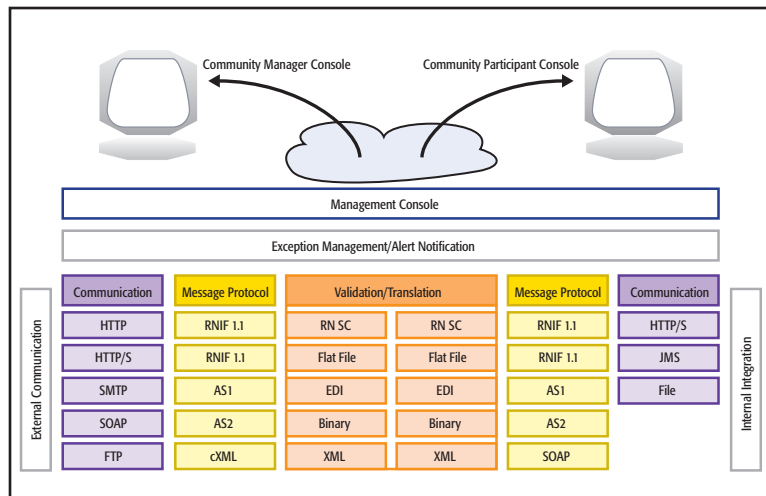


Figure 1 • High-level architecture

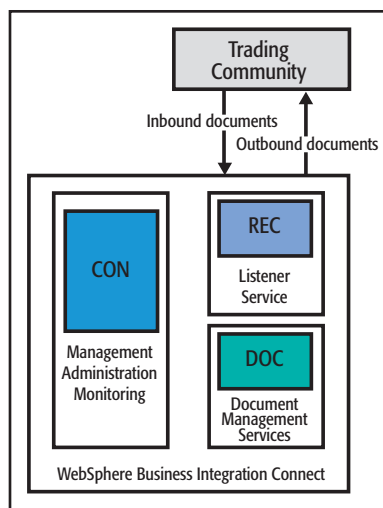


Figure 2 • Components

WebSphere Business Integration Connect allows for validation (using XSD or DTD definitions) and document normalization (via XSLT) integration. After validation and normalization, the document can be processed into other message formats. As the final step, the document is made available for delivery into the hub via HTTP/HTTPS, JMS, or file-based integration. The end-to-end management of the software platform is provided by the event/alert notification layer and the management console, which provides transaction visibility to the hub as well as trading partner participants.

From an architecture perspective, WebSphere Business Integration

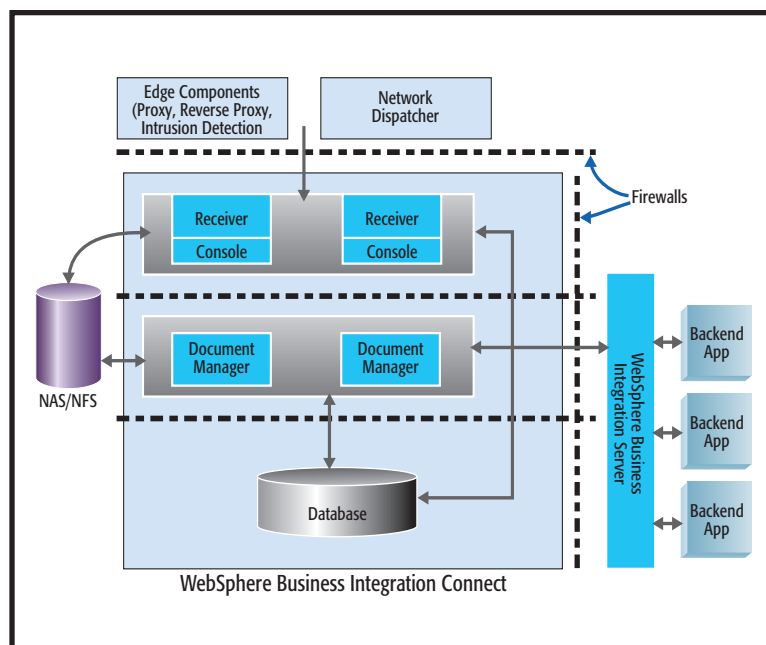


Figure 3 • Split Server topology

Connect consists of three major components:

- **Receiver (REC):** The Receiver provides the inbound message listener service and initiates the document processing flow.
- **Document Manager (DOC):** The Document Manager is the main processing engine and is responsible for document routing, state management, message format packaging, delivery management, and logging, as well as validation and normalization functions.
- **Community Console (CON):** The Console application is a Web-based J2EE application for the operational management of the community hub.

All three components run in separate WebSphere Application Server instances providing container management, as shown in Figure 2. The use of WebSphere Application Server as the foundation for the product enables WebSphere Business Integration Connect components to be deployed on separate servers. This architectural feature facilitates deployments to meet high-end performance requirements as well as support for custom high-availability/failover requirements.

The decoupled architecture is supported through components' interactions with the database, shared file storage, and messaging system. The database holds the community meta-data, validation documents (known as "guidelines"), message history (nonrepudiation), process state management history, and system/document exchange event data. The shared file service is responsible for physical storage and management of the actual documents. The messaging system provides the event service that enables components to communicate with each other. As a result of this component architecture, a Receiver never talks directly to a Document Manager. This decoupled architecture allows an organization to establish multiple receivers and/or multiple document managers depending on load and performance requirements. The ability to implement distributed topologies additionally provides architectural flexibility and allows the product to be configured to support complex, multilayered extranet/DMZ environments. Figure 3 shows a possible configuration using the Split Server topology.

Document security is enabled via encryption/decryption of payloads utilizing PKI standards to secure and validate the authenticity of documents. From a document exchange perspective, WebSphere Business Integration Connect enables the hub to restrict document interactions and SOAP operations to specific trading partners.

From an external access perspective, transport-layer security provides both server-based and client-based certificate authentication at the level of the receiver. Internally, the security integration within WebSphere Business Integration Connect utilizes an access control permission model to configure and enforce user/developer access rights within the Community Console. Partner administrators registered into the system can create user accounts for their organizations as well as groups to provide differential access to specific target roles.

Support for high-availability solutions is fully enabled. Support for highly available Receiver and Console components requires network dispatch services. Document Managers fully support placement on different machines without the need for additional clustering software. Cluster Fault-Tolerant Solutions (e.g., Linux [Red Hat and SUSE] AIX/HACMP, Sun Clusters, and Windows solutions) can be used as well to support high availability. Figure 4 shows the Pod-based solution deployment architecture, which can be used to support active-active configurations to support high-availability functions as well as load balancing.

live wireless.

work wireless.

be wireless.

The most important
technology event
of the year!

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



welcome the wireless generation.

March 22-24, 2004

Georgia World Congress Center

Atlanta, GA, USA

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



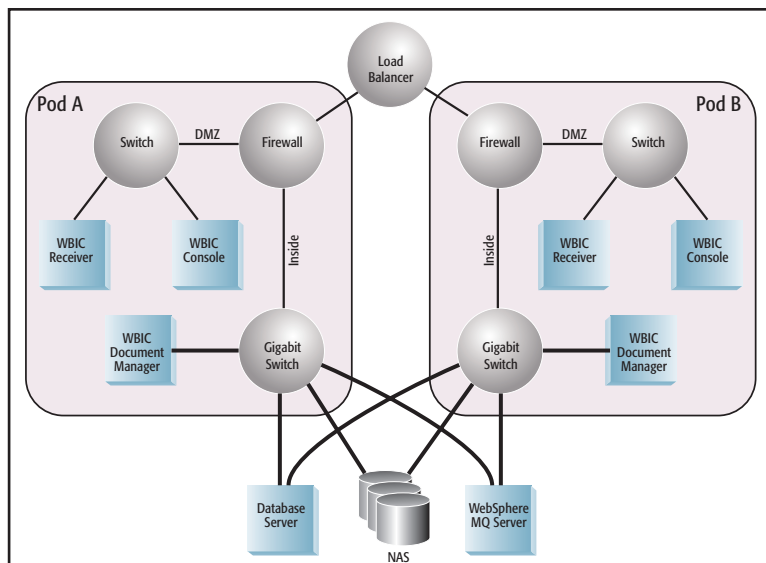


Figure 4 • Pod-based solution deployment architecture

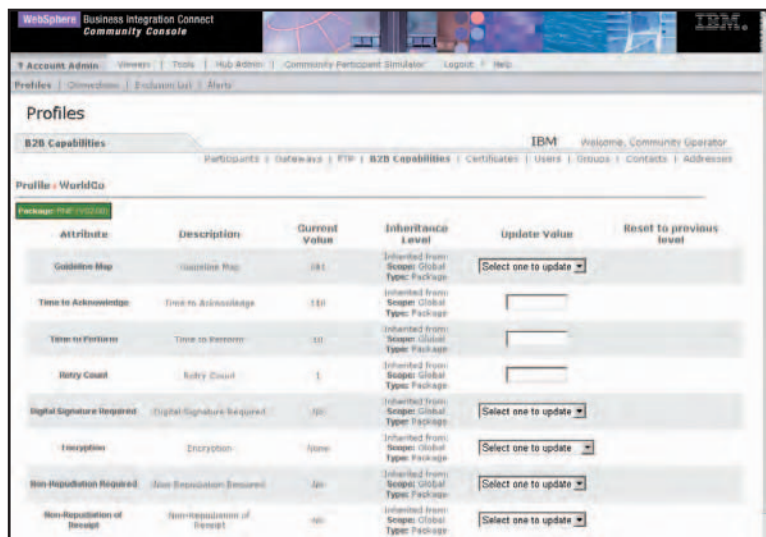


Figure 5 • Web-based Community Console

WebSphere Business Integration Connect enables retry processing at the transport and the process levels. For transport-level retry, the hub will attempt to send a document a configurable number of times with a configurable retry interval between attempts. Failure to complete the message delivery within this set range of attempts results in quiescing the outbound gateway and persisting the messages to a configured gateway message queue. When the gateway is activated, documents flow in a FIFO order from the gateway message queue. Process-level retries occur according to the specific business protocol (such as RosettaNet or AS2) and are configurable by partner and process. Based on a defined number of attempts and a defined retry interval, process retries continue until the message is acknowledged and/or appropriately responded to as per the protocol. In the event of a transport or process-level retry, hub and participants are notified via the alert management facility.

WebSphere Business Integration Connect provides integration with enterprise applications via JMS, HTTP and HTTP/S POST, SOAP, and file-based interactions.

It is optimized to work with WebSphere Business Integration or WebSphere Application Server offerings but is flexible

to work with other middleware products as well.

Organizations can deploy the product with WebSphere Business Integration or WebSphere Application Server offerings. As an example, a new customer order could trigger an event via a WebSphere Business Integration Adapter and the subsequent routing to the WebSphere Business Integration Server. The WebSphere Business Integration Server hosts business processes that drive integration with other enterprise applications as well as interactions with external trading partners. The customer order could, for example, result in an inventory shortage, and an automated interaction with a partner might be needed to provide an available-to-promise date to the customer. During WebSphere Business Integration Server processing, the integration process can issue a request/reply call via WebSphere Business Integration Connect to the supplier.

WebSphere Business Integration Connect also provides a complete solution for EDI integration. Using WebSphere Business Integration Connect for transport integration (e.g., AS2 and FTP), traditional EDI processing can be enabled via WebSphere MQ integration with WebSphere Data Interchange or via file-based/message-based integration with alternative existing legacy EDI transformation packages, e.g., Harbinger/Inovis, Gentran, and others. This approach allows XML-based document exchange solutions to directly co-exist with traditional, time-tested EDI solutions.

Internal integration packaging can be invoked with or without transport-level attributes. Transport-level attributes are used to support additional automated routing of the document. Both HTTP and JMS attributes are formatted/managed as x-aux fields (defined in XML) for message/transport information, e.g., sender_id, receiver_id, process version, protocol, instance_id, and other fields. The WebSphere Business Integration Message Broker provides direct integration to message queues. Integration with WebSphere Interchange Server requires the WebSphere JMS Adapter V 2.3 (or higher) to format/parse the attribute information.

WebSphere Business Integration Connect ships with an extensive set of predefined exception events to enable alerts to be automatically raised during hub operations and notifications to be sent to multiple parties at both the hub and participant levels. From a community integration perspective, the Community Console provides a Web-based interface to provide community visibility into the event history and document exchange interactions. The Community Console gives hub managers an aggregated view of the activities of the community, allowing organizations to pinpoint "weak spots" in the community. Through this Web-based interface, community participants are provided a secure, searchable, Web-based view of their specific exchange activities and provided the ability to update partner-specific information, including profile information as well as participant users and group metadata. The Community Console also provides the interface for the administration of the hub including creation and maintenance of partner profiles, certificate management, and hub administration/configuration tasks. Figure 5 shows a screenshot of the Web-based Community Console.

Summary

Interenterprise integration requirements are fluid and dynamic, which imposes unique requirements on the implementation and operation of trading ecosystems. The need to manage these architectures more proactively is positioning community integration solutions as the core requirement for

WebSphere Data Interchange

In 2001, more than 2 trillion U.S. dollars in transactions were traded via traditional EDI architectures. EDI is established in 95% of Fortune 500 companies, and many of these enterprises have been reluctant to extend this EDI solution into XML-based implementations. As a result, there is a requirement for a robust EDI interchange solution, which is parallel with the WebSphere Business Integration Connect technology.

EDI integration is enabled through the use of WebSphere Business Integration Connect to provide a transport architecture and WebSphere Data Interchange to provide the EDI transformation solution. WebSphere Data Interchange is a data translation application supporting EDI standards (e.g., X12, EDIFACT, UCS, VICS, Tradacoms, and others) and XML formats and operates on multiple platforms including Windows 2000, AIX, and z/OS. The main components of WebSphere Data Interchange include:

- WebSphere Data Interchange Client, which runs on Windows 2000. This is used to create and deploy maps, import EDI standards, and add trading partner EDI profiles.
- The Data Translation application (The WDI "engine"), which transforms application data to EDI (ANSI X12, EDIFACT), XML, and non-EDI formats (user data) providing an any-to-any mapping solution. The Data Translation engine can run on Windows, OS/390, z/OS, and AIX.

building trading communities. Providing event and transactional visibility to the community managers and the community participants, trading community benefits can be realized more rapidly.

WebSphere Business Integration Connect enables community integration regardless of transport, data format, and trust model. With the WebSphere Business Integration Connect technology, customers can scale from a small trading community consisting of 3–5 partners to a complete trading community consisting of tens, hundreds, and thousands of trading partners based on a WebSphere Application Server foundation. The product leverages JMS for WebSphere Business Integration connectivity and offers HTTP as well as file-based integration solutions as well.

Acknowledgments

The author would like to acknowledge the other members of the worldwide product team (including Steve Nowland, Lyle Larson, Doug Hillary, Dave Mulley, Brian Wilson, and many other contributors), the WebSphere Business Integration Connect development team (including Ashutosh Arora, Rayne Anderson, Raja Das, and Pat Keane), and the Viacore team (including Tony Curwan, Jeff Peters, and Eric Nelson). ☺

AUTHOR BIO

Scott Simmons is the worldwide technical lead architect for B2B integration for Worldwide WebSphere Business Integration and is an IBM Certified Senior IT Architect. Scott joined IBM in March 2002 from Peregrine/Extricity and has more than 20 years of experience in the IT industry with companies/organizations including the U.S. Air Force, Atlantic Research, Sybase, and Informix.

SCOTTSIM@US.IBM.COM



THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

Go to www.SYS-CON.com

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

Java
Developer Journal
Industry Newsletter

WebServices
Industry Newsletter

XML
Journal
Industry Newsletter

wireless
Industry Newsletter

WebLogic
Industry Newsletter

WebSphere
Industry Newsletter

ColdFusion
Developer Journal
Newsletter

.NET
Journal
Newsletter

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.

Exclusively from the World's Leading i-Technology Publisher



WRITTEN BY ANJAN MITRA

First-Class Services for the Travel Industry

Managing the quality of Web services systems

AgentWare's Web services-based solutions enable travel agents to locate the best travel options with greater efficiency. By ensuring that these loosely coupled systems are meeting business requirements, AmberPoint Service Level Manager is helping AgentWare continually improve its services.

Low airfares for customers. High-quality customer service from travel agencies. Once, these objectives seemed mutually exclusive. But AgentWare's Web services-based solutions are helping travel agencies bridge the gap. AgentWare gives travel agencies the ability to quickly and easily obtain real-time, panoramic views of fare and schedule information from a single Web interface.

To proactively monitor and manage its subscription-based solution, AgentWare uses AmberPoint's Service Level Manager, which is designed specifically to bring comprehensive service-level management to Web services systems. As a result, AgentWare has been able to exceed customer service expectations, mitigate problems before they can impact business, and develop new revenue opportunities.

AUTHOR BIO

Anjan Mitra is a senior product manager at AmberPoint, Inc. He has 12 years of experience in the software industry and has worked on a broad range of products covering application servers, collaborative development platforms, and e-commerce applications. He has done product management in various capacities at Netscape, AOL, and CollabNet.

found themselves doing manual searches through multiple Web sites to find the best fares for their cost-conscious customers – hardly a recipe for efficiency.

AgentWare solved the problem by enabling travel agents to use a single Web interface to access fare and schedule information from airlines, hotels, and car rental companies. Not surprisingly, travel agencies have welcomed AgentWare's offering with open arms. Over a 14-month period, AgentWare's market penetration within U.S. travel agencies has skyrocketed.

Web Services Only

Because it needed a fast and flexible IT architecture that could easily accommodate this rapid growth, AgentWare built its solutions using only Web services. AgentWare provides access to these services on a subscription basis – customers purchase packages of itinerary and fare requests just as they would buy packages of cell phone minutes. Because AgentWare can spread its infrastructure costs over its entire customer base, it is able to reduce costs to customers.

The Challenges of Subscription Services

Selling subscription-based services meant AgentWare often had to guarantee its offerings through the service-level agreements (SLAs) with which it contracts itself to a combination of uptime, response times, and numbers of concurrent and aggregate transactions. AgentWare also needed to keep track of customer subscriptions and provision services accordingly. Effectively managing its Web services infrastructure was essential to meeting customer expectations.

AgentWare gathers data in real time from external sources, limiting the control the company has over much of its

extra-enterprise system. Nevertheless, they needed to proactively manage potential and unpredictable IT problems, from supplier system downtime to spikes in demand generated by high response to special offers. Whenever it added functionality, it also needed to manage these changes across several different suppliers.

AgentWare considered building its own management functionality, but chose instead to implement Service Level Manager because it offered a prebuilt, industrial-strength system. "Having the AmberPoint system between us and our subscribers suits our business and architectural model by providing real-time application-level monitoring," says David Gruber, CTO at AgentWare.

Differentiated AgentWare Services

AgentWare offers three products, all based on SOAP-compliant Web services, to assist travel agents and corporations that book their own travel: Travel Console, WebPoint, and Data Services. Travel Console, which provides travel agents with a single Web interface for creating searches for flights and fares, goes to outside sources of travel information and returns consolidated flight and fare information to the travel agent in real time. WebPoint integrates this Web fare search and itinerary information directly into Galileo Focalpoint, a widely used product that allows travel agencies to access schedule and fare information, book reservations, and issue tickets for airlines, hotel chains, car rental companies, cruise lines, and tour operators worldwide.

AgentWare Data Services enable large agencies, travel technology companies, and Web sites to integrate the core engine that drives AgentWare's Travel Console and WebPoint products into

their existing travel-related systems. Data Services furnishes both Java- and Microsoft .NET-based gateways to GDS systems, as well as accounting, reporting, point-of-sale, back-office, customer relationship management, and corporate travel management systems.

Decoupling the Management Layer

Because AgentWare must manage so many Web services that it does not control, its management solution could not require any change to the managed applications or to the XML messages. Service Level Manager is built as a Web services intermediary that intercepts Web services traffic to provide the necessary management capabilities, when needed. By abstracting the management from the Web services, AmberPoint allows the application management system to be decoupled from the Web services. This makes it easy for AgentWare to manage pre-existing and external Web services, as AmberPoint's approach does not require additional coding to those Web services.

New Revenue Opportunities

Today, AgentWare leverages Service Level Manager's application-level monitoring to meet – and exceed – its customers' service requirements. AgentWare has also streamlined its operations by using AmberPoint to provision new services, speed development, and track usage trends to proactively head off system problems.

Perhaps the most surprising outcome of AgentWare's use of AmberPoint is that it can leverage detailed information on how customers are using its services to develop new revenue-generating services. For example, says Gruber, "If we see a lot of activity with JetBlue or AirTran, product development might say, 'We need to expand our activity with JetBlue or AirTran by developing a service that enables travel agents to check their commissions with those airlines.'"

Better Customer Service

By providing targeted, valued-added services for customers, AgentWare enjoys greater repeat business. For example, because it tracks the subscription for each customer, AgentWare can proactively offer renewals or additional services to customers who are approaching the usage limits specified in their contracts. It helps customers to avoid service interruptions by alerting them when they are about to bump up against these limits. AgentWare also uses the information provided by

AmberPoint to feed their billing system.

AmberPoint's reporting capabilities allow AgentWare to provide customers with useful business information as well. For example, AgentWare can use Service Level Manager to furnish select customers with custom reports about their airline bookings. Whereas previously individuals within a particular company might have purchased airline tickets on an ad hoc basis, the AgentWare reporting service allows travel agents to determine the extent to which the entire company uses each airline's services. With this information, the company might choose to negotiate volume discounts with the airline.

Customer service representatives can configure a personalized portal view to display information about specific services, alerts, agreements, reports, customers, and business dashboards. These views can incorporate custom dashboards with charts, numbers, graphs, and so on, that can be targeted to very specific monitoring requirements. These dashboards can be created and incorporated in the Service Level Manager portal without any programming, allowing each end user to interact with the UI in a manner that is most suited to his or her responsibilities.

AmberPoint provides the ability to rapidly assign and remove services based on subscriptions for new and existing AgentWare customers. Additionally, AmberPoint integrates with a system AgentWare had previously implemented to authenticate each client. If the requester is not a valid customer then the request is blocked or returned with an appropriate message.

Faster Troubleshooting

AgentWare has streamlined its development and testing processes as well. Prior to using Service Level Manager, AgentWare stored all data about XML traffic in either a database or a log file. But it had no easy way to identify or locate the particular message or content that developers thought might be causing a problem on the network. Developers had to export data from the database and manually comb through the log files.

Service Level Manager's informative alerting and reporting capabilities make it easy for developers to find the information they need in order to troubleshoot their systems more quickly and easily. For example, says Gruber, "When we sign up new customers, we can give them an AgentWare test environment with AmberPoint running between their systems and our services. When the cus-

tomers call and says, 'We sent you a message 30 minutes ago and we're not sure we sent it right,' AmberPoint Service Level Manager helps us easily pinpoint the customer, the corresponding SLA, and the Web service that was affected. By providing us with detailed impact analysis, logs, timely alerts, and online views, the product helps us rapidly identify and actively address existing, or even potential, problems."

These capabilities have dramatically improved the effectiveness of AgentWare's troubleshooting efforts. Says Gruber, "AmberPoint has reduced the time it takes to troubleshoot by 25%."


Correlating System and Application Data

AgentWare's data center uses Service Level Manager to correlate system information, such as network traffic, TCP/IP bandwidth utilization, and CPU utilization, with what's going on in the application at a business level. By providing a context for system-level network traffic, Service Level Manager helps the AgentWare data center better manage its network, as well as proactively plan for future growth. For example, AgentWare can now determine whether usage spikes are due to increased business or network problems that are causing the system to send duplicate messages about the same root problem.

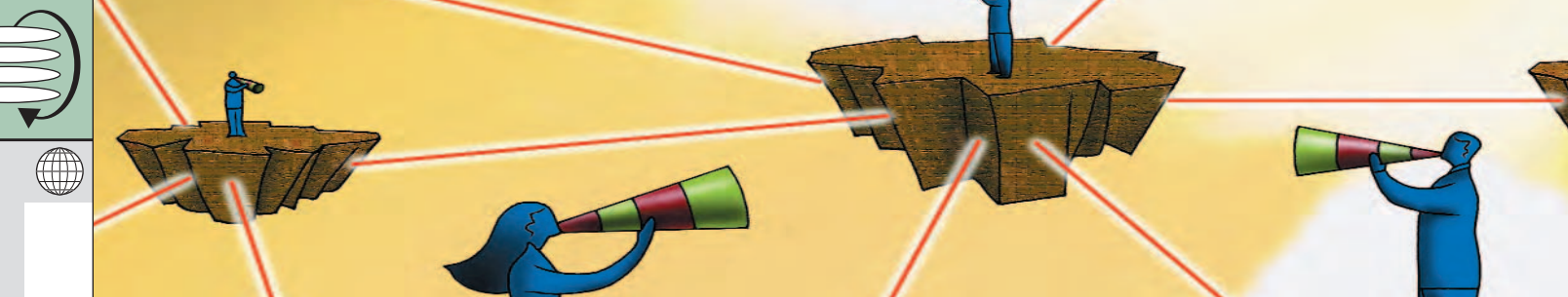
Proactive Capacity Planning

Data center managers use AmberPoint to proactively address potential problems in delivering service levels before they occur. For example, AmberPoint might indicate that AgentWare's network is receiving an extraordinary number of extra requests for JetBlue or AirTran during a certain time frame. With this information AgentWare might choose to increase the capacity of the network link to these airlines before problems occur.

Service Level Manager capabilities such as e-mail alerts, which automatically contact operations managers when the application is encountering problems, also help AgentWare to maintain system uptime and performance to better comply with its service level agreements.

"The bottom line," says Gruber, "is that the AmberPoint Service Level Manager provides technology that allows us to provide innovative new business services to our customers, better manage our service, and operate more efficiently and effectively." 

AMITRA@AMBERPOINT.COM



Troubleshooting .NET Applications

Implementing an XslTransform wrapper to trace and profile XSLT transformations Part 1

WRITTEN BY

KARTHIK RAVINDRAN

In this article I will show you how to implement an XslTransform wrapper to trace and profile XSLT transformations at runtime in .NET applications. The source code and supporting files can be downloaded from www.syscon.com/xml/sourceec.cfm.

Tests executed and results obtained to verify the execution and performance of XSLT transformations during development and testing are often not fully representative of production use case scenarios encountered in a live environment. Troubleshooting XSLT transformations that performed well during the development and testing phases but later exhibit unexpected behavior in a production environment can be a daunting task.

Random exceptions, access violations, generation of unexpected/incorrect output, and inconsistent drops in query execution performance (prolonged high CPU usage or idle times) are examples of common problems that you can encounter when executing XSLT transformations in production environments. Such problems are most commonly caused by the design of the XSLT style sheet(s) used to execute the transformations. These problems can be difficult to reproduce and troubleshoot when the source XML data is generated dynamically and varies by user, e.g., dynamically generated queries to retrieve source XML data from a SQL Server back end, or data retrieved dynamically from external Web services.

In such situations the availability of the following diagnostic data can help significantly reduce the time to identify the cause of the problem in an XSLT transformation and take the required corrective action:

1. The context source XML data
2. The XSLT style sheet used to execute the transformation
3. An XSLT transformation trace/profiler log to the point of failure, manual termination, or completion

The context source XML data can be difficult to obtain in dynamic environments without source code modifications. The availability of this item combined with the style sheet (which is usually easily attainable, as style sheets are commonly loaded from the disk) will provide users the data they will need to recreate the problem on a development/test server.

An XSLT transformation trace/profiler log will help execute a time- and resource-efficient post mortem analysis of the failed/problematic XSLT transformation to determine the progress to the point of failure/manual termination/completion, isolate the cause of an observed problematic behavior, and take the required corrective action.

These items should be obtainable without having to resort to executing time-consuming troubleshooting steps that require the use of advanced debugging tools and/or the imple-

mentation of source code modifications. Being able to obtain diagnostic data representative of a live use case in a timely manner will facilitate faster problem isolation and resolution.

In the .NET Framework, the XslTransform type in the System.Xml.Xsl namespace implements the API to execute XSLT transformations using the .NET XSLT processor. This article details an implementation architecture that can be used to implement a wrapper for this type to trace and profile XSLT transformations at runtime in .NET applications.

XSLT Tracing Goals

Design Goals

1. The tracing functionality should be fully configurable without requiring source code modifications. This is mandatory if a feature like this is to be useful in production environments. Specifically, the following aspects should be configurable:
 - Enable/disable tracing. By default XSLT Tracing should be disabled. The feature is meant to be enabled and used only in troubleshooting and/or profiling scenarios.
 - Configure (enable/disable) XSLT elements/instructions to be traced, e.g., trace only xsl:template instructions.
 - Enable/disable the persisting of source XML data used in XSLT transformations. This option, when enabled, will help obtain the source XML data transformed by problematic XSLT transformations in dynamic environments.
 - Enable/disable the deletion of trace logs and source XML data persisted by successful XSLT transformations (to prevent logs related to successful transformations from cluttering the disk).
 - Specify the physical path of the disk and folder where the trace log and source XML data should be persisted.
2. There should be no requirement to mix trace points and tracing code with the core query processor implementation. Users will not have access to the source code of the XslTransform type and will therefore not be able to instrument its implementation to achieve the desired tracing.
3. It should generate a trace log with useful data that can be easily interpreted by an XSLT developer. Third-party application profilers that profile managed code execution in .NET applications are available. The data generated by these profilers reflect the execution and performance of API calls executed on the .NET assemblies used during the course of the transformation. A user will need to be familiar with the internal XslTransform implementation to map its API calls to corresponding XSLT instructions in a style sheet. This level of expertise and skill is currently possessed only by the Microsoft developers who implemented the XslTransform type. An XSLT transformation trace log should reflect the execution of the

style sheet using a format that is independent of implementation specifics and interpreted by an XSLT developer.

User Goals

1. Enable users to easily obtain live source XML data in dynamic environments in order to recreate and troubleshoot XSLT transformation failures.
2. Eliminate the need to install/configure advanced debugging tools to troubleshoot XSLT transformation problems and failures.
3. Enable experienced XSLT developers to efficiently and independently troubleshoot and resolve .NET XSLT transformation problems. The feature can also be used to obtain and provide Microsoft product support services with live data to reproduce an observed problem when encountering a potential anomaly that requires confirmation.
4. Enable inexperienced XSLT developers to efficiently gather and provide Microsoft Product Support Services with the data (source XML + style sheet + trace log) required to troubleshoot the cause of an observed .NET XSLT transformation problem.

Non-Goals

1. High performance when XSLT tracing is enabled is not a design goal. Tracing functionality of any kind is meant to be used only in troubleshooting/profiling scenarios. ODBC tracing is a perfect example of this. When enabled, it significantly slows down the execution of ODBC applications. XSLT tracing is no different, and its benefits come with an associated performance trade-off.
2. Tracing will be configurable only for specific XSLT instructions. Tracing every XSLT instruction is not required to isolate the cause of an XSLT problem in the vast majority of the use cases. The ability to trace a few core XSLT instructions (e.g., xsl:template, xsl:for-each, and xsl:value-of) should be sufficient to aid in the problem isolation process. Additionally, the requirement to trace each supported XSLT instruction should be configurable.

Architecture

Figure 1 is a high-level diagram of the architecture that will be used and implemented by the XslTransform wrapper to trace XSLT transformations in .NET applications. The following aspects of the architecture will help achieve the design and user goals.

1. Application configuration files will be used to configure XSLT tracing. This is the preferred approach to be able to dynamically configure the aspects of XSLT tracing described in the design goals without requiring source code modifications. This flexibility comes at the cost of the configured settings having an application-wide impact. In this context, when XSLT tracing is enabled, all XSLT transformations executed by an application will be traced. When it is disabled, none of the XSLT transformations will be traced.
2. The code to generate the trace log will be implemented in a CLR type. When tracing is enabled, object instances of this type will be instantiated and supplied as extension objects to trace XSLT transformations.
3. When the option to trace XSLT transformations is enabled, the XSLT processor will instrument XSLT style sheets to add the required trace points. The trace points in the instrumented style sheet will be extension object method calls to the tracing extension object. Each extension object method call will supply parameters that reflect the following:

- XSLT instruction to be traced
- Values specified in the original style sheet for the common attributes of the XSLT instruction being traced, e.g., the

value specified for the selected attributes of xsl:value-of and xsl:for-each instructions

- The context node processed by the instruction (when applicable)
- The context node data (when applicable)

The tracing extension object method will log these parameters in a tab-delimited text file, thereby generating an execution trace log as the transformation is processed and executed. For each instruction, the time elapsed since the execution of the previous traced instruction and the total cumulative time elapsed since the transformation began will also be reported. These values will serve as simple but useful profiling data that can help identify performance bottlenecks in the style sheet execution.

Instrumenting the style sheet to add trace points eliminates the need to have access to the query processor's implementation to implement tracing functionality. The goal is to treat XSLT as an XML transformation/query programming language that can be instrumented and whose execution can be traced like any other regular programming language.

4. The generated trace log can be opened and viewed using Microsoft Excel. This will contain an instruction trace to the point of failure, manual termination (hang scenarios), or completion. In failure and manual termination scenarios, the trace log can be used to study the execution sequence of the transformation and isolate the cause of the problem. In scenarios where the transformations run to completion, the profiling information in the trace log can be used to evaluate the performance and identify any significant bottlenecks that may exist (see Figure 2).
5. When the option to persist the source XML is enabled, the XSLT processor will write the source XML to a dynamically generated XML document that will be persisted in the trace output folder. The persisted source XML can be matched with its instruction trace log using the timestamp value used to construct the names of these files.
6. When tracing is disabled, there will be no performance hit and the XSLT processor will execute XSLT transformations in a regular mode as depicted in Figure 3. This is very impor-

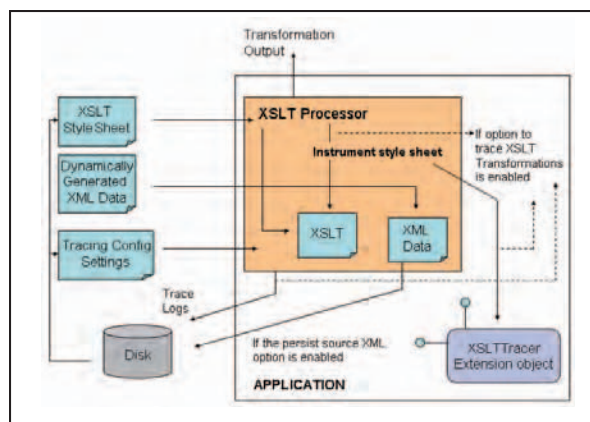


Figure 1 • Architecture of the XslTransform wrapper

	Instruction	match	select	Context node	Node value	Elapsed time
1						
2						
3	xsl:template	OrderDetailsData	N/A	OrderDetailsData	N/A	0:0:0.0
4	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0.0
5	xsl:value-of	N/A	OrderID	Orders	10248	0:0:0.0
6	xsl:value-of	N/A	OrderDate	Orders	1996-07-04T00:00:00.0000000-04:00	0:0:0.0
7	xsl:for-each	N/A	following-sibling::OrderDe	OrderDetails	N/A	0:0:0.0
8	xsl:value-of	N/A	ProductID	OrderDetails		11 0:0:0.0
9	xsl:value-of	N/A	Quantity	OrderDetails		12 0:0:0.0

Figure 2 • A snapshot from a trace log generated by the XslTransform wrapper

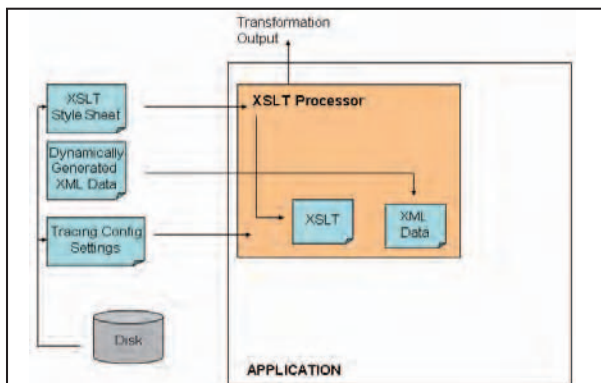


Figure 3 • When tracing is disabled, the XSLT processor will execute XSLT transformations in a regular mode

tant to ensure that there is no additional overhead when XSLT tracing is disabled (the default).

Implementation

Note: The implementation discussed in this section and provided as a download with this article is an unsupported sample. Do not use it directly in production environments. The objective of this implementation is to illustrate how tracing functionality can be implemented to trace XSLT transformations in .NET applications. You are encouraged to study the source code of the sample XslTransform wrapper, add additional features as required, and use it in development/testing environments to troubleshoot and profile XSLT transformations executed by your .NET applications. You are also encouraged to invest additional development and testing efforts to implement/enhance and fully test an identical custom XslTransform wrapper for use in your production environments.

This section will explain the implementation of the sample XslTransform wrapper to trace and profile XSLT transformations in .NET applications. The implementation will be based on the architecture detailed in the previous section. An understanding of the design goals and the architecture is required to comprehend the implementation specifics.

Prior to getting started, you will first need to download and set up the sample project files provided with this article. Execute the following steps to do this:

1. Download TracingSample.zip from www.sys-con.com/xml/sourcecec.cfm and save it on your local hard disk.
2. Extract the contents of the zip file to the root folder of your hard drive (C:\). This will create two subfolders, named TracingDemo and TraceOutput. The XsltDiagnostics solution in the TracingDemo\XsltDiagnostics folder contains the Visual Studio .NET 2003 project files for the XslTransform wrapper class and the XSLT tracing extension object. The TracingClient solution in the TracingDemo\TracingClient folder contains the project files for a sample console EXE client application that will be used to illustrate the XSLT tracing functionality implemented by the XslTransform wrapper. TraceOutput is configured as the XSLT tracing output folder in the TracingClient application configuration file.

Open TracingClient.sln (located in the TracingDemo\TracingClient folder) and XsltDiagnostics.sln (located in the TracingDemo\XsltDiagnostics folder) in two instances of the Visual Studio .NET 2003 IDE. Enable the option to Show all files in the Solution Explorer window of both instances.

With the sample files set up, we will now see how the architecture described in the previous section maps to the implemen-

tation of the sample XslTransform wrapper.

The XSLT Processor

The sample XslTransform wrapper type will be used as the XSLT processor in this implementation. The XsltDiagnostics.XslTransform2 type implements this wrapper. Wrapper implementations are required in order for the XslTransform constructor, the Load method, and the Transform method to implement the desired tracing functionality. To keep things simple and to be able to focus on the implementation of the tracing functionality, the XslTransform2 type implements wrapper methods for only the following Load and Transform method overloads:

```
Load(string url)
Transform(IXPathNavigable,XsltArgumentList,TextWriter)
```

Don't worry about understanding the functionality of the code in these wrapper methods at this point. I will get into this as we dig deeper into the implementation specifics of the tracing functionality. For now it is sufficient that you understand that the code written to use the XsltDiagnostics.XslTransform2 wrapper type to execute XSLT transformations (using the supported Load and Transform wrapper overloads) will be identical to the code required to use the System.Xml.Xsl.XslTransform type for the same purpose. The following code taken from the Main method of the TracingClient.StaticClient type illustrates this:

```
XslTransform2 stylesheet = new XslTransform2();
stylesheet.Load("people.xml");
XPathDocument data = new XPathDocument("people.xml");
System.IO.StreamWriter output = new
System.IO.StreamWriter("output.html");
stylesheet.Transform(data,null,output);
```

The XSLT tracing functionality is configured using an application configuration file (discussed next) and used by the XslTransform2 Load and Transform wrapper methods. No source code modifications or setting of additional properties is required in the client applications to trace XSLT transformations executed by them. Subsequent references in this article to the term XSLT processor refer to the XsltDiagnostics.XslTransform2 type and/or object instances of it.

Configuring XSLT Tracing

In this implementation, regular .NET application configuration files will be used to configure XSLT tracing. For EXE applications, this will be the <AppName>.exe.config file located in the executable folder, and for ASP.NET applications it will be the Web.config file. Keys to configure the aspects of tracing described in the design goals will be defined in an <appSettings> section. The XslTransform wrapper will read in these configuration settings on instantiation to determine the XSLT tracing options that must be applied when executing transformations.

Listing 1 is a sample <appSettings> section (taken from the TracingClient application configuration file) used to configure XSLT tracing in this implementation. (All of the code referenced in this article can be downloaded from www.sys-con.com/xml/sourcecec.cfm.) The use of the configuration options is described in Table 1.

On instantiation the XSLT processor will read in these settings from the application configuration file. The settings specified for each of these configuration parameters will control the subsequent tracing functionality.

Listing 2 shows the code in the constructor of the XsltDiag-

Configuration option	Use	Default value
TraceXSLTStyleSheets	Used to enable/disable XSLT tracing in an application. Tracing should be enabled only when there is a requirement to troubleshoot an XSLT transformation.	False
XSLTTraceOutputPath specify an existing	Used to specify the output folder where the trace logs and source XML data (if the XSLTTracePersistSourceXML option is enabled) should be persisted.	No default. The user must specify an existing folder.
TraceXSLTTemplates	Used to enable/disable the tracing of xsl:template instructions.	True
TraceValueOf	Used to enable/disable the tracing of xsl:value-of instructions.	True
TraceForEach	Used to enable/disable the tracing of xsl:for-each instructions.	True
XSLTTracePersistSourceXML	Used to enable/disable the persisting of source XML data used in XSLT transformations. This option should be enabled only when there is a requirement to capture dynamically generated source XML data required to recreate an observed problematic behavior.	False
XSLTTraceDeleteLogs WhenSuccessful	Used to enable/disable the deletion of trace logs and persisted source XML data for successful transformations. This option should be disabled only when using the tracing functionality to profile the performance of successful transformations.	True

Table 1 • Configuration options

Parameter	Applicable to
Name of the XSLT instruction	xsl:template, xsl:for-each, and xsl:value-of
Pattern specified for the match attribute	xsl:template
QName specified for the name attribute	xsl:template
Expression specified for the select attribute	xsl:for-each and xsl:value-of
Local name of the context node	xsl:template, xsl:for-each, and xsl:value-of
The context node	xsl:template, xsl:for-each, and xsl:value-of

Table 2 • The tracing extension object method parameters

Instruction	Position of trace point in relation to instruction	Example
xsl:template	First child	<pre><xsl:template match="EMPLOYEES"> <xsl:value-of select="XSLTTrace:traceInstruction ('xsl:template','EMPLOYEES', 'N/A',local-name(.),.)" /> </xsl:template></pre>
xsl:for-each	First child when there is no child xsl:sort Second child when there is a child xsl:sort	<pre><xsl:for-each select="EMPLOYEE[DEPT='IT']"> <xsl:sort select="NAME" /> <xsl:value-of select="XSLTTrace:traceInstruction('xsl:for- each','N/A','EMPLOYEE[DEPT=&quot;IT&quot;]' , local-name(.),.)" /> </xsl:for-each></pre>
xsl:value-of	Replaces instruction	<pre><xsl:value-of select="translate(NAME,'o','O')"/> will be replaced by <xsl:value-of select="XSLTTrace:trace Instruction('xsl:value- of','N/A','translate(NAME, &quot;o&quot;,&quot;O&quot;')',local- name(.),.)" /></pre>

Table 3 •The position of a generated xsl:value-of trace point in relation to the XSLT instruction being traced

nostics.XsltTransform2 type to read in the settings specified for the XSLT tracing configuration parameters. The code in the constructor instantiates the wrapped System.Xml.Xsl.XsltTransform object, and uses an instance of the System.Configuration.AppSettingsReader .NET Framework class to read and store the values of the XSLT tracing configuration parameters defined in the application configuration file. The configuration parameter names are case sensitive (by virtue of being specified as XML). An attempt to read a missing parameter (not defined in the configuration file, misspelled, or typed using the wrong case) will generate an InvalidOperationException. Each GetValue method call is wrapped in a distinct try...catch block to prevent a missing/misspelled configuration parameter from hindering the attempt to read subsequent configuration parameters. The TraceXsltStyleSheets parameter is used to determine whether or not to enable XSLT Tracing. Tracing will be disabled when this parameter is set to false (the default) or is missing/misspelled.

Instrumenting the XSLT Style Sheet

XSLT style sheet instrumentation is used to add trace points to an XSLT style sheet to trace its execution. In the sample implementation, the instrumentation process can be configured to add trace points to trace the execution of the following XSLT instructions. Tracing these instructions should help you narrow down on the problematic style sheet segment/instruction:

1. xsl:template
2. xsl:for-each
3. xsl:value-of

You can extend the implementation to trace the execution of additional XSLT instructions using the implementation technique described in this section.

When the style sheet is initially loaded, the value of the TraceXsltStyleSheets configuration parameter is used to determine whether or not it should be instrumented. Trace points will be introduced in the style sheet to trace its execution when this option is enabled (disabled by default). When disabled, the original style sheet is loaded as is with no modification.

The trace points in the instrumented style sheet are calls to a tracing extension object method that takes the parameters shown in Table 2 (as applicable to an instruction being traced) and uses them to generate the execution trace log. The instrumentation routine parses each XSLT instruction for which tracing is enabled to extract the values that must be supplied for the match, name, and select parameters. The local-name XPath function and the period (.) context node XPath operator are used to supply the last two parameters.

The tracing extension object method will use these parameters to generate a tab-delimited execution trace log as the transformation is executed. For each instruction, the time elapsed since the execution of the previous traced instruction and the total cumulative time elapsed since the transformation began will also be reported. These values are close approximations and will serve as simple but useful profiling data that can help identify performance bottlenecks in the style sheet execution.

xsl:value-of elements are used to insert the extension object method calls to trace the style sheet execution. The extension object method will return an empty string for xsl:template and xsl:for-each instructions after logging their execution. For xsl:value-of instructions, it will evaluate the select expression and return the result as a string. This makes it possible to completely replace an xsl:value-of instruction with a call to the extension object method (see Table 3).

The sample implementation does not automatically resolve and instrument style sheets imported (xsl:import) and/or included (xsl:include) by the main style sheet loaded

into the XSLT processor. The source code for the sample implementation can be downloaded from www.sys-con.com/xml/sourcec.cfm. Study the inline comments to understand the implementation specifics.

Executing the Style Sheet and Tracing the Transformation

When an XSLT transformation is executed with the tracing option enabled, an object instance of the XSLTDiagnostics.XSLTTracer type is instantiated and associated with the context of the XSLT processor. The XSLTDiagnostics.XSLTTracer type implements the functionality of generating the XSLT execution trace logs. As explained in the previous section, the trace points in the instrumented style sheets are calls to the traceInstruction method implemented by this type. This method is invoked by the XSLT processor for each configured instruction as the transformation is executed. Suitable parameter values required to generate a trace line for each instruction will be passed to the method when it is invoked.

Listing 3 shows the code used to implement the Transform wrapper method in the XsltDiagnostics.XsltTransform2 type. Study the inline comments to understand its functionality.


The code to persist the XML data is implemented in the private RecursivelyWalkAndPersistData method. You can examine this method at your convenience to see how code can be written to persist data loaded into an IXPathNavigable object that does not implement a Save method (e.g., XPathDocument).

The generated trace log and the persisted XML data will be deleted if the option to delete logs for successful transformations (XSLTTraceDeleteLogsWhenSuccessful) is enabled. The default setting for this parameter is True to prevent the logs generated for successful transformations from cluttering the disk. When profiling the performance of XSLT transformations that execute to completion you should disable this option to be able to view the generated trace log.

Note: When tracing is disabled, the transformation will be executed in a regular mode without accruing any of the overheads associated with tracing its execution.

Conclusion

Instrumenting and tracing XSLT transformations by treating XSLT as a regular programming language can provide very useful diagnostic data to help analyze and troubleshoot related problems. The sample implementation discussed in this article is a proof of concept "scratch the surface" prototype to illustrate how this can be done and the value of XSLT tracing. In Part 2 of this series I will walk you through a couple sample scenarios to illustrate the use of the XSLT tracing implementation discussed here.

I hope that I have managed to whet your appetite on this subject and get your creative juices flowing on ways to further enhance this base implementation to address additional specific requirements that you might have. I am also very interested in hearing your feedback on the value of a feature like this being built into the .NET System.Xml and MSXML stacks. 

AUTHOR BIO

Karthik Ravindran is a technical/product lead in the Microsoft Web Data PSS group. His core responsibilities are helping enterprise customers successfully implement solutions using WebData XML technologies and working closely with the product group on delivering product feedback and inputs on the implementation of future upcoming related technologies to address top customer issues and feature requests.

This article originally appeared in .NET Developer's Journal

XMLSEMF@MICROSOFT.COM

For more detailed information, please visit:
lighthouseseminars.com or call **Joe Richard** at **781.821.6734**

**Learn about successful
implementation of content
technology through the
unique combination of**

- Case Studies
- Project Management Techniques
- New Technology Trends

**Register
NOW!**

Register now for the **CONFERENCE PLUS PACKAGE**
and receive a **FREE iPod!**

LIMITED QUANTITY, REGISTER EARLY!

For details go to **lighthouseseminars.com**



Platinum Sponsors



Gold Sponsors



Analyst Sponsor



Analyze the Future

Media Sponsors



Association Sponsors



OPEN SOURCE CONTENT MANAGEMENT



The Gilbane Conference

ON CONTENT MANAGEMENT

March 24-26 2004

Los Angeles, California

The Westin Bonaventure Hotel & Suites





Matters of Syntax

ConciseXML builds upon the important qualities of XML and S-Expressions

ConciseXML is a new syntax I co-developed with Christopher Fry that builds on the best features of XML and S-Expressions while eliminating their constraints.

XML originated with the document-markup world of SGML and has become the leading syntax of the Internet. Its use has been for documents and data – not for programming logic.

S-Expressions, or symbolic expressions, is the syntax behind Lisp-like languages, including *Scheme*. Basically, S-Expressions are nested lists of symbols. S-Expressions are used with languages that support the notion that code is data.

The many discussion postings on the Web regarding these topics would indicate a disconnect between the S-Expression and XML camps. The purpose of this article is to bridge the two worlds and offer a solution that addresses everyone's needs.

To illustrate the differences among the three syntaxes, let's start with a data structure expressed in XML, S-Expressions, ConciseXML, and also a commonly used semicolon-delimited syntax used by many programming languages, including Java. While there are multiple ways of encoding the data in XML and S-Expressions, only one is shown for the purposes of the first comparison.

• XML

```
<book isbn="0764525360" title="Water Language" copyright="2002"/>
```

• S-Expressions

```
(book "0764525360" "Water Language" 2002)
```

• ConciseXML

```
<book "0764525360" title="Water Language" copyright=2002/>
```

Syntax vs Language

A syntax is structural only. It defines the construction of valid expressions, not whether those expressions have a semantically valid meaning. For example, XML is a syntax that does not associate meanings with symbols or tag-names.

A language uses symbols, which are assigned specific meanings. For example, HTML is a language because the tags have associated meanings. XHTML is a language that uses XML syntax.

• Semicolon-Delimited Syntax

```
new Book("0764525360", "Water Language", 2002);
```

There are a number of similarities among the first three syntaxes:

- The expressions are delimited by characters that indicate the start and end of the expression. S-Expressions are delimited by a set of parentheses “()”, while XML and ConciseXML expressions use angle brackets “< />”.
- The expression name occurs immediately after the initial delimiter.
- White space is used to separate arguments.
- The syntax is independent of a specific language.

The semicolon-delimited syntax is different from the others in several ways. A comma is used between arguments, and there is no set of delimiter characters that always begin or end an

expression. Parsing a semicolon-delimited syntax requires that the parser have language-specific knowledge. The first three syntaxes can be easily parsed, independent of any language.

Now let's look at some of the differences among the four syntaxes.

Representing Arguments

With XML, the arguments (XML attributes) must include a key to identify the argument name.

Basic S-Expressions syntax does not support argument keys, and there is no one standard for adding keys, although there are several different ways people have attempted to add keys to S-Expressions.

With ConciseXML, arguments have an optional key. Therefore the arguments can be either keyed arguments or unkeyed arguments. The previous example shows an unkeyed ISBN argument, while the title and copyright arguments are keyed.

Unkeyed or By-Position Arguments

While XML requires the use of keyed arguments, the syntax of most programming languages, including S-Expressions, does not support keyed arguments. All arguments must be passed by position. The example below compares a call to a `createBook` method in a semicolon-delimited syntax used by languages such as Java, C#, or C++ with the same call in S-Expressions syntax.

• Semi-Colon Delimited Syntax

```
createBook("0764525360", "Water Language", 2002);
```

• S-Expressions

```
(createBook "0764525360" "Water Language" 2002)
```

AUTHOR BIO

Mike Plusch has over 10 years of industry experience building platforms and complex Web applications for organizations including Digitas, Harlequin, and Bowstreet, a pioneer in Web services. He is the co-inventor of the Water language and ConciseXML. He has developed a broad range of applications for financial services, travel, retail, and manufacturing. An accomplished author, Plusch has written two books and contributed to numerous books and articles on Web services, including *Water: Simplified Web Services and XML Programming*, Wiley; and *Water Programming*, Mike Plusch and Christopher Fry. Plusch holds two degrees from MIT, one in management and one in computer science.

Now
More
Than
Ever
Before...

JAVA DEVELOPER'S JOURNAL

JDJ

Means
Business



Subscribe Now!

\$69⁹⁹
1 YEAR/
12 ISSUES

www.SYS-CON.com/JDJ

or call

1-888-303-5282

The World's Leading i-Technology Magazine Just Got Better!

Now with expanded business coverage, *JDJ* is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of IT professionals.



**The Most Popular
i-Technology Magazine
in the World !**

**SYS-CON
MEDIA**

The World's Leading i-Technology Publisher

Definition of Content Argument

XML can contain a non-empty element or an element with a content area. The term “empty element” refers to a call expression without content such as:

```
<STUFF/>. The content argument is delimited with a special syntax familiar to HTML users: <STUFF> this is the content </STUFF>.
```

XML requires a key for every argument. The workaround for this restriction is to use the content area of an XML expression for argument values. Here is a typical example of that encoding technique in XML:

```
<books>
  <book isbn="0764525360" title="Water Language"/>
  <book isbn="0972006702" title="Water Programming"/>
</books>
```

Unlike most programming languages, XML does not have one standard method for representing an array or a generic ordered collection. Consider an ordered data structure consisting of an integer number, a decimal number, and a string. An array representing that data structure might appear as the following in several languages:

```
[5, 10.3, "stuff"]
```

There is no standard form to express this in XML because XML does not have a standard “array” or “vector” element and by-position arguments are not supported. In ConciseXML syntax, this example would be represented with unkeyed arguments and “vector” for the expression name in a similar style as S-Expressions.

- **ConciseXML**

```
<vector 5 10.3 "stuff"/>
```

- **S-Expressions**

```
(vector 5 10.3 "stuff")
```

By allowing both keyed and unkeyed arguments, ConciseXML supports the by-position passing of arguments used in the syntax of almost every program-

ming language, as well as the keyed arguments required by XML and HTML.

Ordered Collections

Most programming languages have a data structure that associates a key with a value. This structure is typically named a hash table, dictionary, or associative array. The value of the data can be any object, and typically the key can also be any object. With XML, argument keys are restricted to strings, and therefore many data structures are difficult to represent. For example, integer keys are not permitted in XML. In ConciseXML, however, integer keys are represented as shown in the following expression:

```
<vector 0=5 1=10.3 2="stuff"/>
```

“One of the advantages of S-Expressions and ConciseXML is that they use a single standard method for representing nested data”

Argument Value

An *argument value* is the value of the argument, and does not include the argument key. Let's take another look at the first set of examples:

- **XML**

```
<book isbn="0764525360" title="Water Language" copyright="2002"/>
```

- **S-Expressions**

```
(book "0764525360" "Water Language" 2002)
```

- **ConciseXML**

```
<book "0764525360" title="Water Language" copyright=2002/>
```

- **Semicolon-Delimited Syntax**

```
new Book("0764525360", "Water Language", 2002);
```

In all four syntaxes, the value of the ISBN argument is the string “0764525360”. The value of the “copyright” argument, though, is the string “2002” in XML and the integer 2002 in the other three syntaxes.

In XML, every argument value must be delimited by quotes, which implies that the value must be a string. S-Expressions and ConciseXML permit argument values to be any expression,

not just strings. For example, the value of the copyright argument is the integer 2002, not the string “2002”. For representing data, it is important that the value of an argument can be anything, not just a string. The integer value 2002 is very different than the string “2002” that contains four characters. The ability for an argument to hold any value becomes even more important when representing nested data. One of the advantages of S-Expressions and ConciseXML is that they use a single standard method for representing nested data. Because the argument values may be any expression, an argument, such as “next_edition”, could have a value that is another call expression. This makes it easy to have a book as the value of a field within another book:

- **ConciseXML**

```
<book isbn="0764525360" next_edition=<book isbn="0764525361"/> />
```

- **S-Expression**

```
(book "0764525360" (book isbn="0764525361") )
```

Although this data can also be represented in XML, there are multiple ways to encode the same data. Here are three valid XML encoding styles:

- **Style 1**

```
<book isbn="0764525360">
  <next_edition><book
```

Definitions

One of the difficulties in comparing various syntaxes is that they use different terminology to refer to similar concepts. This article uses the following core definitions.

- **Expression:** A syntactically valid chunk of text.
- **Call expression:** An element or tag in XML, an expression in S-Expressions.
- **Expression name:** The area beginning a call expression. In the previous example, the expression name is “book”.
- **Argument:** XML calls these attributes. Arguments have a value and may have a key. S-Expressions do not have argument keys, while XML requires keys for every argument.

ConciseXML Makes Eight Extensions to XML 1.0

1. *Attribute values can be any expression:*

```
<input size=3/> or
<person birth=<date year=2002
month=10 day=2/>/>
```

XML requires all attributes values to be quoted, effectively requiring all values to be type string. Elements are often used to work around this limitation, presenting another set of problems.

2. *Attribute keys can be any object:*

```
<thing 0="foo" <date 2004 10
10/>="mplusch"/>
```

XML does not let an attribute key start with a digit or contain angle-brackets. That effectively limits attribute keys to strings. ConciseXML makes it possible to easily represent array-like fields with integer keys as well as any object by using a ConciseXML call syntax.

3. *Tagname of an element can be any expression:*

```
<foo.bar/>
```

XML namespaces are a step in this direction, but ConciseXML makes it possible to use any expression as the tagname of an element. The tagname may be a path or a call/tag.

4. *Attribute keys are optional:*

```
<date 2004 month=10 day=28/>
```

In the CSV (comma separated value) syntax and in all major programming languages, field or argument values are given by position, not by keyword.

5. *Closing tagname is optional:*

Not only does this remove unnecessary clutter, but when ConciseXML is used as the syntax for dynamic languages, the tagname may not be known until runtime, therefore the closing tagname must be optional.

6. *Top-level can be any expression, not just an element:*

For example, true. It is surprisingly difficult in XML 1.0 to create a document whose value is a simple type such as a string, number, or boolean value.

7. *Multiple top-level expressions:*

The CSV file format and most programming languages allow multiple top level expressions. XML 1.0 allows only a single root element in a file. ConciseXML permits any number of expressions at the top level.

8. *Attribute type:*

In addition to a key and a value, attributes can also have an optional type that is delimited by an equal sign:

```
<thing
some_key="some_value"=some_type/>
```

sometimes referred to as self-describing. However, because of the ambiguity in representing non-string data, XML documents require additional information about the encoding style used. XML, therefore, is not a self-describing syntax.

On the other hand, S-Expressions and ConciseXML support argument values that can be any expression. Complex data structures can be easily represented and do not require the use of a content argument. The content argument offers an important feature, though, for representing content that contains both text and other expressions.

Let's take a simple example from HTML:

```
<H1> A heading with <B>bold</B> text
</H1>
```

The example shows the content of the H1 expression comprised of a sequence of hypertext. The three expressions are the string "A heading with", the expression bold, and the string "text". This use is extremely common in HTML and very useful for mixing text and data.

S-Expressions do not have a convenient way to represent mixed text and data; therefore, S-Expressions do not have the ability to handle a very common feature of markup languages.

S-Expressions can easily handle nested data structures, but do not support keyed arguments or the content argument for integrating text and data. XML supports the content argument and keyed arguments, but does not easily represent complex data structures or unkeyed arguments. ConciseXML represents an important milestone in the development of a common syntax because it supports the best features of both S-Expressions and XML.

...

This article has introduced a few of the features of ConciseXML that build upon the important qualities of XML and S-Expressions, namely optional keys, argument values that can be any expression, and conveniently representing mixed text and data. To learn more about ConciseXML and how it extends XML by eliminating eight constraints of XML, please visit www.ConciseXML.org. For a free trial of Steam XML software, a platform that makes use of the Concise XML syntax, visit www.clearmethods.com.

MPLUSCH@CLEARMETHODS.COM

```
isbn="0764525361"/></next_edition>
</book>
```

• Style 2

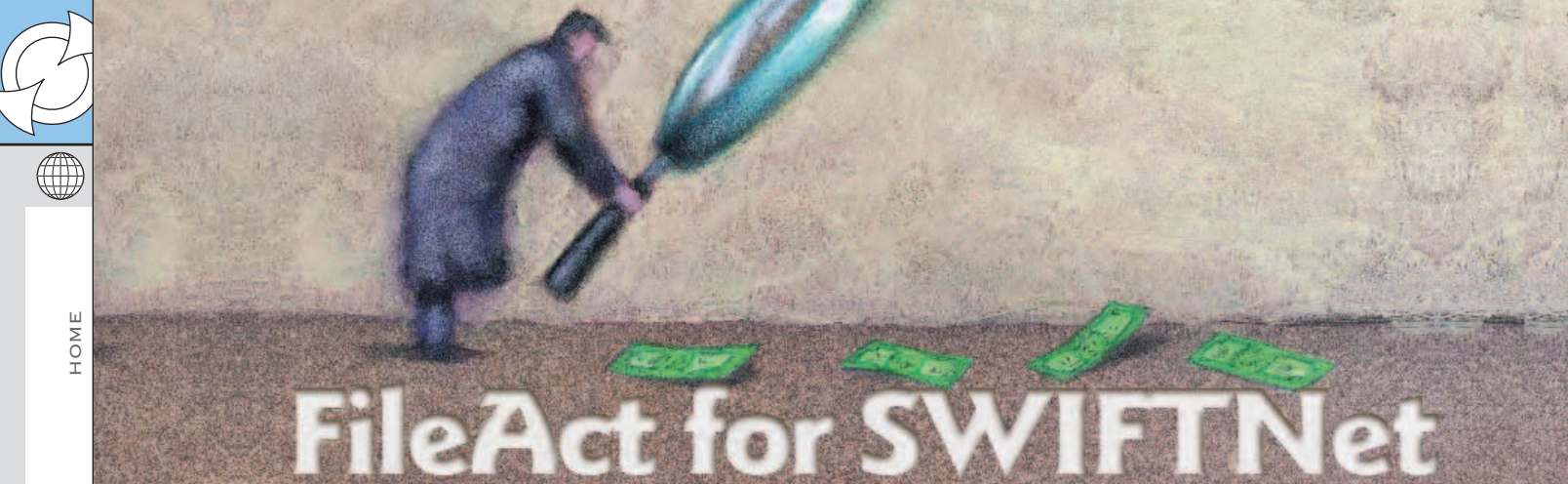
```
<book isbn="0764525360">
  <field key="next_edition"><book
isbn="0764525361"/></field>
</book>
```

• Style 3

```
<book isbn="0764525360">
  <attr><key>next_edition</key>
    <value><book
isbn="0764525361"/></value>
  </attr>
```

```
</book>
```

Because XML argument values must be strings, the content area of an XML expression must be used to represent non-string argument values. XML does not have a standardized representation for nested data, which introduces significant ambiguity when interpreting the precise meaning of an XML expression. This ambiguity causes major problems when XML syntax is used to transport data between entities without an explicit agreement on a particular encoding style for XML syntax. XML is



FileAct for SWIFTNet

WRITTEN BY

CATHY KINGETER

Helping financial services gain savings from their network

As in any vertical industry, there are business drivers in financial services that require automated and reliable movement of data between applications and between organizations. Repetitive credit transfers such as pension or salary payments, ACH processing, securities clearing and settlement instructions, reports sent to regulatory bodies, and cash management are a few examples. Maximizing profit on every exchange is gained by moving the transactions closer to real-time processing.

Automated transactions are often routed through the S.W.I.F.T. (Society for Worldwide Interbank Financial Telecommunications) network, an industry-owned cooperative supplying secure messaging services and interface software to 7,000 financial institutions worldwide.

SwiftML, or the S.W.I.F.T. Markup Language, is an XML vocabulary used to automate Swift messages. The design of SwiftML accounts for the interoperability issue between different financial XML implementations through the use of SWIFT-Standards Modeling. This new standardization approach allows the separation of the business standard from its physical representation in XML syntax. All this information is stored in the SWIFTStandards Repository. This allows different languages, even those that are non-XML, to interoperate through the use of the common business model. SwiftML is in line with major XML standardization initiatives (e.g., ebXML, a global initiative to achieve common and predictable usage of XML cross-industry). Consistent and uniform SwiftML messages come from the structured approach of SwiftML adoption. For example, business information is always expressed as XML elements/values; metadata information is expressed as XML attributes.

While many financial industry IT decision makers are familiar with S.W.I.F.T.'s comprehensive messaging standards, security, reliability, and common platform of advanced technology, fewer may be intimate with the benefits and advantages of SWIFTNet's FileAct for mission-critical file exchange. For financial institutions looking for methods of leveraging an investment already in place with SWIFTNet, moving files is a primary avenue to efficiency and savings since it's considerably less expensive than messaging.

FileAct enables secure and reliable transfer of files and is ideal for exchanging batches of structured financial messages, large reports, images, etc. It also supports targeted solutions for market infrastructure communities, closed user groups, and financial institutions. While FileAct can support any file type, applications that are proving popular include bulk payments, securities value-added information, and reporting.

Several progressive banks, such as Wachovia Corporation, are already subscribing to FileAct and by doing so are seeking to maximize opportunities provided by FileAct, Browse, Interact, and the Closed User Group concepts offered by S.W.I.F.T. FileAct is actually a fairly straightforward "out-of-the-box" service. Once registered and the necessary SWIFTNet connectivity and interface structure is in place, exchanging files can begin almost immediately, with the required connectivity and interface infrastructure similar to SWIFTNet FIN and other SWIFTNet services.

For forward-looking financial institutions, the eye is on getting FIN traffic moved to meet the S.W.I.F.T.-mandated migration schedule. The looming migration is from the S.W.I.F.T. FIN network, an X.25 physical network, to SWIFTNet FIN, a secure Internet-type network. All current users of the FIN network are required by S.W.I.F.T. to migrate to SWIFTNet FIN no later than December 2004, with country and volume windows already established for the migration. The global implementation of SWIFTNet and SWIFTNet FIN continues the enhancement and integration of S.W.I.F.T.'s family of messaging services, enabling more interactivity, store-and-forward and file transfer messaging, and secure browsing capabilities. (Figure 1 shows the file transfer interface, and Figure 2 shows the remote adapter interface.)

In addition to FileAct, S.W.I.F.T.'s InterAct includes both an interactive transmission – the ability to query another participant and receive an immediate response – and, eventually, the replacement of SWIFTNet FIN for the transmission of single transaction transmissions as required by the sender and underlying business function. The Browse capability offered by the global implementation of SWIFTNet and SWIFTNet FIN allows clients or designated entities to enter a data domain and extract information within identified parameters under specified circumstances.

On top of such advancements and enhancements, S.W.I.F.T. recently announced a comprehensive set of price reductions for FileAct, InterAct, and Browse as well as SWIFTNet FIN prices – the biggest overall price-reduction initiative since 1996, when FIN prices were cut by 30%. The reductions include a 45–67% price reduction for both InterAct and FileAct for domestic traffic. Value-added features such as nonrepudiation, message priority, store and forward, and delivery notification will be unbundled.

To encourage wider adoption of S.W.I.F.T. standards and messaging, S.W.I.F.T. has introduced new packages that substantially lower the entry barriers and complexity for smaller banks and financial institutions that connect to S.W.I.F.T.

Some of the pricing initiatives include dropping several up-front fees for software and security.

In addition to channeling main network traffic, another critical ROI factor is extending the value proposition of S.W.I.F.T.'s suite of services to participants' back-end systems. With S.W.I.F.T.'s introduction of "alliance gateways," there is a need to consolidate back-end traffic from such gateway points of presence on the network – a huge consolidation effort.

Some third-party network products, such as Sterling Commerce's Connect:Direct for SWIFTNet, integrate the delivery and receipt of files with the back-end applications – the real consumers of the data. Connect:Direct for S.W.I.F.T. interoperates with FileAct and integrates seamlessly with back-end applications. It also can provide decreased pricing for SWIFTNet usage, enhanced by efficient compression.

The advantages of SWIFTNet's FileAct and its complementary services along with recent price reductions give banks and financial institutions an effective way to squeeze more efficiency and savings from their SWIFTNet migration investment and SWIFTNet infrastructure. By combining a reliable network solution that integrates the back end with points of presence, an organization is well on the way to staying competitive in a challenging and ever-changing financial services market.

SWIFT in Action

Wachovia Corporation is one of the largest full-service financial services firms in the United States. The company provides a broad range of banking, asset management, wealth management, and corporate and investment banking products and services. For a number of years, Wachovia has partnered with banking entities in Europe to provide national pension payment services in North America and Latin America. Wachovia receives the pension payments in bulk files. Upon receipt, the bulk payment data is broken down by Wachovia internal applications into individual transactions that are then processed and disbursed via multiple delivery methods, including ACH, EFT (Canada), Fedwire/CHIPS, book credit, and check.

Historically, Wachovia used the S.W.I.F.T. Integrated File Transfer (IFT) product for receipt of bulk payment data over the S.W.I.F.T. X.25 network. With the decommissioning of the IFT product by S.W.I.F.T., and the migration to the IP-based SWIFTNet, Wachovia found itself in need of a viable replacement solution. This solution would have to support the new S.W.I.F.T. FileAct Service, provide nonintrusive integration with their back-office applications for payment disbursements, and support future applications.

A Winning Combination


Wachovia decided to deploy a combination of Connect:Direct for SWIFTNet and Connect:Direct – both by Sterling Commerce – because of its automation capabilities, management, security, and reliability features. Connect:Direct also provided Wachovia the flexibility they needed to support their back-office systems involved in trade services and payment processing, including AS/400 and Tandem.

Connect:Direct made the integration of existing back-office applications with SWIFTNet FileAct Service simple. Connect:Direct for SWIFTNet provided all the base capabilities of the SWIFTNet FileAct Service through its command set to enable the following file exchanges: Sending a file – a request to send a file to a counterparty over SWIFTNet; Getting a file – a request to retrieve a file from a counterparty over SWIFTNet; Receiving a file – respond to a file transfer request by a counterparty that wishes to send a file over SWIFTNet; and Downloading a file – respond to a file transfer request by

a counterparty that wishes to download a file from the local SWIFTAlliance Gateway over SWIFTNet.

Connect:Direct enabled Wachovia to automate the business processes involved in moving files bidirectionally between various internal applications and the SWIFTAlliance Gateway as well as sending and receiving files via the SWIFTNet FileAct Service in conjunction with correspondent banks.

With the combination of Connect:Direct and S.W.I.F.T.'s FileAct Service, Wachovia enjoys improved bandwidth over current IFT implementation; support for any file type; automated, integrated business process that embraces back-end applications; decreased pricing for SWIFTNet usage enhanced by efficient compression; and single protocol with SWIFTNet.

The combined solution also provides Wachovia a data exchange infrastructure to support existing and future business applications. These applications could extend beyond the data now exchanged with correspondents to include other payment types, settlements, image transfers, trade documents, consolidated statements, and securities-related information. 

AUTHOR BIO

Cathy Kingeter is solutions manager for S.W.I.F.T. at Sterling Commerce.

CATHY_KINGETER@STERCOM.COM

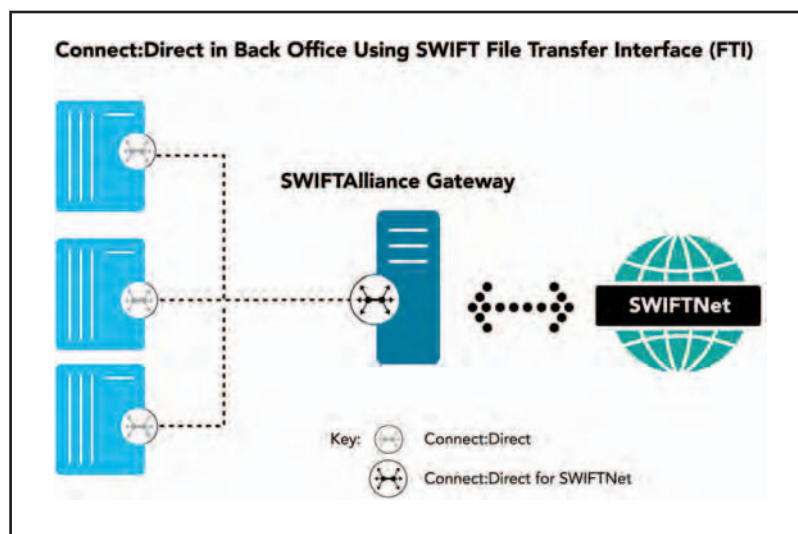


Figure 1 • File transfer interface

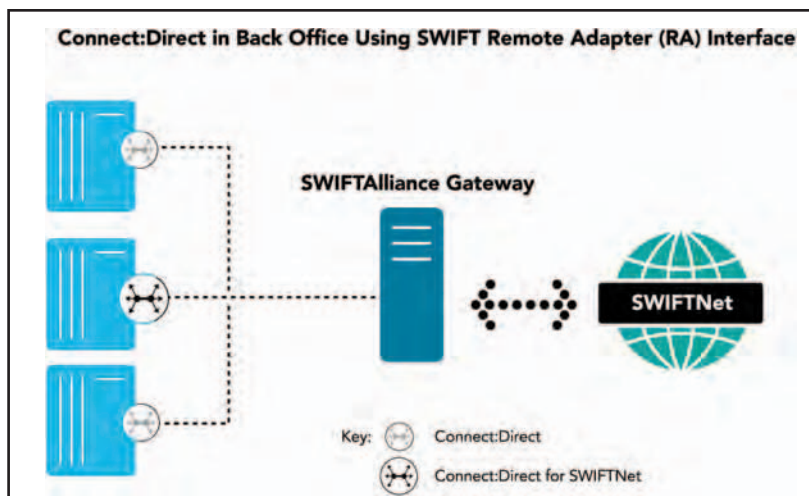
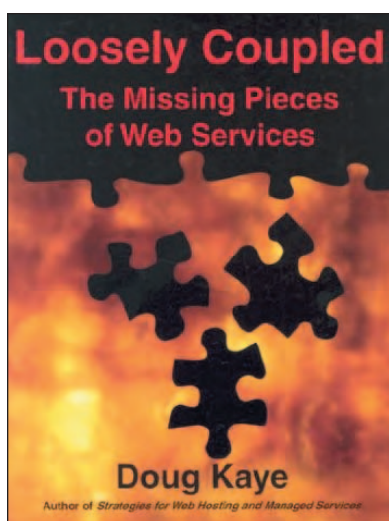


Figure 2 • Remote adapter interface

Web-Services Transactions

From Loosely Coupled – The Missing Pieces of Web Services



Most non-programmers think of transactions as associated with buying and selling, credit-card authorizations, and the like. But in the jargon of computer science, the word *transaction* has a very specific meaning: the interaction and managed outcome of a well-defined set of tasks. If that definition still sounds rather vague or abstract, it's because the scope of what's considered a transaction has expanded over the past two decades, and the older simpler definitions are no longer adequate. Computer systems have been connected via networks, and applications are more distributed in nature. The theories and practices of transactions have been repeatedly stretched to their limits, re-evaluated, and extended. Now, because of web services, we're once again expanding that definition to include long-lived loosely coupled asynchronous transactions.

Transaction Basics

Most database operations are sim-

ple, and thus don't qualify as transactions per se. For example, when a customer-service application wants to look up a customer's phone number, the application sends a query message to the database. It's a read-only operation that involves only one record in a single database. But most importantly, it's a one-step (atomic) operation that doesn't interact or conflict with other applications that may be interacting with the same record or even the same database.

More complex database operations require multiple steps that must all be completed for the operation to succeed. We refer to these operations as transactions. The traditional definition of a transaction is a single unit of work composed of two or more tasks. If any of these component tasks cannot be completed, the entire transaction fails, leaving the data in the state it was in before the transaction was initiated. In other words, a transaction is a collection of tasks that either all succeed, or all fail. Achieving this consistent termination of a unit of work is the goal of a traditional transaction-processing monitor (TP monitor) which is software that manages lower-level database operations.

An example of a simple transaction is a transfer of funds from one account to another within the same bank. The transaction's unit of work consists of two tasks: the debiting from one account, and the crediting to another. Ideally, both tasks will execute properly (commit), but even more important is that if one task can't be accomplished, neither will be executed (i.e., they'll both abort). It's okay if the matching credit and debit both fail—the application initiating the transaction can always try again. But it's a serious problem if the credit is executed without the associated debit, or vice versa.

ACID

As the results of their theoretical studies of transactions, Theo Härder and Andreas Reuter published a 1983 paper, "Principles of Transaction-Oriented Database Recovery," in which they presented the requirements for systems that could process multiple-task units of work (transactions), and would not be corrupted by hardware, database, or operating-system failures. The paper is most famous for its specification of the principles of Atomicity, Consistency, Isolation, and Durability (ACID). A system that conforms to these so-called ACID properties guarantees the reliability of its transactions.

Two-Phase Commit

When all of the data involved in a transaction resides on a single database, only one TM is required to maintain atomicity. But applications and databases are increasingly distributed, such as those linked by web services. The challenge for web services is to maintain atomicity by guaranteeing the mutual success and durability of all of the elements of such a distributed transaction, so named because it involves a distributed unit of work. In other words, multiple steps are required that involve two or more databases.

The traditional method for handling distributed transactions is known as the two-phase commit, which, as its name implies, breaks transactions into two cooperating phases. The two-phase commit protocol is illustrated in Figure 1.

The two-phase commit process assures the atomicity of the distributed transaction. It's clean and simple—except when things go wrong. Due to hardware, software, or communications failures, it's possible that one or more

WRITTEN BY DOUG KAYE



AUTHOR BIO

Doug Kaye is the CEO of RDS Strategies LLC and the publisher of the IT Strategy Letter.

This article is excerpted from *Loosely Coupled – The Missing Pieces of Web Services* (ISBN 1881378241). ©2003 Doug Kaye.

messages may be lost, resulting in an uncertain state for one or more of the resource managers. As it turns out, however, only the loss of a commit message can cause a serious problem. Losses of other message types are less critical. If a resource manager fails to get the request-to-prepare message, it will simply fail to respond. The controller will give up waiting for the resource manager's response and send out an abort message. The other resource managers will not have committed any of their changes. The same occurs if one or more of the response messages is lost. And if a done message is lost, no action need be taken, since all of the resource coordinators will have committed the transaction.

The most serious problem occurs when a resource manager prepares for the transaction but never receives either a commit or an abort message from the transaction coordinator. Once a resource manager has sent its prepared response, it's in limbo. It can't commit the transaction, and it can't release any resources locked on behalf of the transaction. (Resource locks are under the control of the individual resource managers, not the transaction controller.)

In fact, there's no simple solution to this problem. No two-phase commit protocol can protect against all failures. The possibility will always exist that a communications failure can cause a resource manager to become blocked, or unable to commit or abort. Still, even with its limitations, the two-phase commit protocol remains the mainstay of distributed transactions.

The Web-Services Challenges

The ACID model has been the focus of transaction technologies for twenty years. It's widely used for both local and—via the two-phase commit protocol—distributed transaction systems. But as valuable as the ACID model has proven to be for tightly coupled distributed systems, it falls short for long-lived, loosely coupled asynchronous transactions.

Long-lived transactions

Web services are far more complex in terms of time and space than the transactions for which the ACID concepts were developed. Whereas ACID-based transactions may span many seconds or even a few minutes, loosely coupled web-services transactions may extend over hours or even days. Considerable time can elapse between the preparation and commit phases. Using ACID-

style transactions in such long-running business processes would mean that participating resources could be locked and unavailable for extended periods of time—which is unacceptable to many local applications that use the same databases and pend until the resources they require are released.

Reliability

ACID-style transactions are designed to cope with failures in hardware, software, and communications, but only in otherwise reliable environments where such failures occur relatively infrequently. Most ACID-style distributed transactions systems are based on synchronous, connection-oriented protocols, which maintain communications paths between transaction coordinators and the participating resource managers for at least the duration of the transaction. These synchronous protocols assist in handling such errors by signaling the transaction-coordinator or resource-manager software when a communication failure occurs, so that the coordinator or resource manager knows it can no longer communicate with the service at the other end of the connection. When a communications link fails, all synchronous transactions that depend on that link are promptly aborted.

Short-term communications failures are therefore fatal errors for tightly coupled synchronous transactions, but they must be routinely handled by the systems that support long-lived, loosely coupled asynchronous transactions. The latter are based on a reliable-messaging infrastructure that delivers messages with a high degree of assuredness, even in cases where the recipient and the intervening infrastructure may be down for extended periods of time.

Trust

Because the resource locks typically used with ACID-style transactions may block applications, it's critical that they be held for as short a time as possible. If an application dies after locking a resource, that resource could be orphaned forever. If the resource in question represents the availability of an airline seat, that seat might never be filled. A resource manager therefore manages its resources like a mother hen, making sure that locked resources are never abandoned. If a local application requests a lock and then terminates, the resource manager must clean up the mess by unlocking the resource. Before a resource manager allows transactions to be initiated by remote transaction coor-

dinators, a great deal of trust must exist among the resource manager, the remote coordinators, and other resource managers participating in the transactions.

Suppose it's not the link that fails, but rather the remote transaction coordinator. Although the messaging software won't signal a communications error (the communications link is still operational), the local resource manager has the ultimate fallback: It can rely on timeouts to protect its resources. Unfortunately, timeouts can't be used for long-lived transactions, because by definition they execute over extended periods. Again, the techniques that support ACID-style transactions won't work with those that are long-lived, loosely coupled, and asynchronous.

Cancellation risks and abuses

External web services introduce a number of risks just by exposing internal systems to access by others. Allowing externally initiated transactions increases what's known as cancellation risk. For example, consider airline seats purchased at full price a few months before the flight. If they're cancelled at the last minute, the airline may be unable to sell them.

The problem becomes more acute when business processes are automated by web services, because accidental or even intentional abuse can so easily go undetected. For example, imagine how an unethical travel aggregator might exploit an airline-reservation web service. Months in advance, the aggregator reserves every available seat on a particular flight—but at the last minute, cancels them. In a panic to sell the seats, the airline puts them on sale at a deep discount. The unscrupulous travel aggregator then repurchases the same seats at this much lower price.

Accepting a reservation carries an

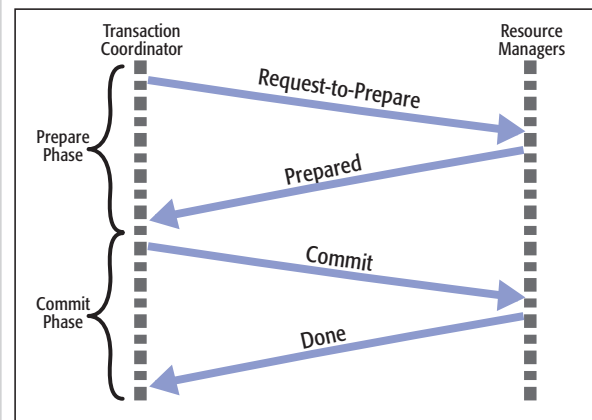


Figure 1 • The two-phase commit protocol

inherent risk of such a last-minute cancellation. This problem exists even without web services, but there are systems in place to detect and prevent most abuses. Airlines manage this risk through overbooking. Concert and theater ticket agencies protect themselves using no-refund policies. But many other businesses – particularly those in wholesale trade – have no formal methods for managing cancellation risks. The risks and abuses of cancellations will probably increase and spread to other industries as external web services are deployed. Web services will ultimately need to express and negotiate the policies under which such transactions are made.

Loosely Coupled Transactions

Clearly, the web-services requirements for transactions far exceed what can be accomplished using traditional technologies. The more loosely we couple systems – separating them in time, space, and control – the more difficult it becomes to manage transactions distributed among them. Loosely coupled transactions, it would seem, come at a cost of increased complexity. That's true, but only so long as we keep trying to apply, refine, and improve traditional approaches based on ACID-style concepts. Instead, let's consider how we can build an all-new transactional system based on loosely coupled web services technologies: asynchronous communications, reliable messaging, and document-style interaction. Let's use an example of a tightly coupled transaction, then see how it can be improved.

You're in your car, listening to the radio, when you hear an announcement that your favorite musician will be performing in your town. You grab your cell phone and dial the ticket-sales agency. A friendly salesperson answers the phone, and you launch into your request – only to be interrupted by the salesperson telling you, "I'm sorry, but our computers are down right now, and we don't know when they'll be back up. You'll have to call again later."

You've just stumbled into one of the drawbacks of synchronous transactions: In this case, there's nothing you can do but abort the transaction. You (the requestor) and the reservation system (the provider) must be available simultaneously. There's no point leaving your information with a salesperson who's just an intermediary, with no store-and-forward capability. Even if the salesper-

son were willing to take down your information, would you trust that person to complete your order? The responsibility for recovering from the system failure and restarting the transaction falls entirely on you, the requestor.

Half an hour later, you call back (retry), and learn that the system is now available. Of course the context of your transaction has been lost, so you've got to start from the very beginning. As luck would have it, the agent submits your request only to report, "Sorry, but all of the orchestra seats are now sold out. The best I can do is row J, seats 103 and 104 in the upper mezzanine." For a period of a few minutes, the reservation system locks the database records that represent those two seats while you make up your mind. If other customers are placing orders through different agents, they won't be offered those same seats. (This is now a synchronous transaction.)

“Clearly, the web services requirements for transactions far exceed what can be accomplished using traditional technologies”

You tell the agent you'll take the tickets, but your cell phone goes dead just as you're about to jot down your confirmation number. Now what? Did the transaction complete? Do you really have two tickets for the concert, or do you need to call back and place another order? If you do, will you end up with four tickets instead of two? Unfortunately, there's no way to know. Such are the problems of tightly coupled transactions without a reliable asynchronous messaging infrastructure.

Wouldn't it be great if you could just leave a voice-mail message (a self-contained document) including not only the obvious details, but instructions (the business logic) for what to do in case your first-choice seats aren't available? Your voice-mail message would then enter a message queue along with those of other customers, and be processed in sequence. As a result of your request, the ticket agency would call you back or send you an email message confirming your purchase. The acknowledgement

would complete this long-lived, loosely coupled asynchronous transaction.

Long-lived transactions

By communicating asynchronously, you've eliminated the real-time constraint of the transaction. You can make your request in the middle of the night. Even if a human agent must review your order, that person need not be available at the time you submit it. Although the vendor's voice-mail system must be able to accept calls at a reasonable rate, the actual transaction system that processes the request is highly scalable. Even if the transaction system goes offline, all orders will get processed in due time as long as customers can submit voice-mail orders. You can see how a reliable asynchronous messaging system is key to long-lived, loosely coupled asynchronous transactions.

Isolation without locking

You've also eliminated the need for record locking. So long as all requests are submitted through a single queue, the ticket agency can process its requests serially. And provided only one ticket request is being processed at a time, the application doesn't need to simulate serialization by locking resources.

Compensating Transactions

Once a transaction has been committed, it can no longer be aborted. Yet in the real world, there are often times when the effects of a transaction must be undone. The problem is that some transactions can't be reversed because their effects are permanent, and/or conditions have so changed over time that restoring the previous state would be inappropriate. As an example, consider a transaction that triggers the manufacturing of an item. Materials are consumed, and money is spent. It's impossible to simply wipe out the transaction. You can't un-manufacture the item. Instead, other actions must be taken, such as charging the customer a cancellation fee and offering the item for sale to other parties. In the earlier example of an unscrupulous travel aggregator who cancelled airline tickets at the last minute, we saw how the airline chose to put those seats on sale at a discount in order to make sure they'd be sold and the airplane would be full.

These are examples of compensating transactions that can be applied after an original transaction has been commit-

ted in order to undo its effects, without necessarily returning resources to their original states. Many transaction managers support compensating transactions – and as we'll see in the case of long-lived, loosely coupled asynchronous web services, compensating transactions can actually be used instead of resource locking.

Optimism

ACID-style transactions are optimistic, and assume a high likelihood of success. You can imagine a human coordinator of a simple two-phase commit transaction commanding the participants. Phase One: "Okay, here's what you need to do. [Coordinator enumerates the requirements.] Has everyone prepared for the transaction by safely storing the results? Good." Phase Two, after receiving affirmative votes from all participants: "Now everyone...GO!" There's no need for the coordinator to ask whether anyone was unsuccessful, since all of the participants promised in Phase One that they could do as requested. The key to the success (and integrity) of the transaction is the locking of the resources

between these two phases.

On the other hand, a loosely coupled transaction coordinator must take a pessimistic view of a transaction's outcome. Even with a reliable messaging protocol, many other errors can occur due to the long-term nature of the transaction. Rather than reserve their resources in advance, loosely coupled participants prepare compensating transactions that will undo the local effects in case the first phase is unsuccessful. If the transaction is later aborted, all participants execute their compensating transactions.

When using compensating transactions, our human coordinator might say in Phase One, "Okay, here's what you need to do. Don't do it yet, but in case this doesn't work, I want each of you to figure out ahead of time how to recover. Now everyone...GO!" Then, in Phase Two: "Great...did that work for everyone, or do we all need to run our back-out scenarios?"

Compensating transactions are one of the technologies that decouple systems from one another, and are a first step towards filling in the missing pieces of complex web services.

Standards

IBM, Microsoft, and BEA are at work on WS-Coordination, a framework that supports multiple coordination types including WS-AtomicTransactions for short-lived "all-or-nothing" transactions, and WS-BusinessActivity for long-lived loosely coupled transactions using compensation.

Sun, Oracle, Iona and others have announced plans for WS-CAF, the Web Services Composite Application Framework, for transactions and coordination of interdependent web services. And the OASIS Business Transaction Technical Committee is continuing to develop BTP, the Business Transaction Protocol, but they're awaiting implementations so that they can progress it towards a full OASIS standard.

The issues are both political and technical. Because the traditional mechanisms for handling distributed transactions don't work for web services, the standards for web-services transactions will be some of the last to be developed, agreed to, and adopted. Most experts don't expect much impact from these competing standardization efforts until 2005.

DOUG@RDS.COM



SPECIAL

*The Web Services Journal–
XML-Journal Readers' Choice Awards*
The "Oscars of the Software Industry" recognize
the best in XML and Web services



XML & .NET

Troubleshooting .NET Applications
Part 2: Sample scenarios illustrating the
use of XSLT tracing



FEATURE

*The Power of Smart Documents in
Word 2003* Let your knowledge workers
focus on what they know best – the content



FEATURE

BI Light
XML for business intelligence

DON'T MISS XML-J MARCH

REAL-WORLD SOLUTIONS



Remember ebXML?

Doing business in real time

While there are many standards that look like ebXML, ebXML is the first horizontal standard designed to address the exchange of information and adherence to inter-enterprise processes. However, in attempting to reach this lofty goal, ebXML is also a complex standard and takes some understanding before we can comprehend its value to the world of application integration and electronic business.

So, why do we need the ebXML standard? It's really a matter of leveraging the Internet to automate how we do business in real time, leveraging common processes and common information formats. The use of an electronic information standard and enabling technology drive how we do business, and the interest in standards that provide a common mechanism to do this will only push us farther along.

Thus was born ebXML, a collaboration between UN/CEFACT and OASIS. As you can tell by the name, ebXML is built on top of XML, as well as other Internet standards, including Web services, to create an infrastructure for information-based and process-based electronic businesses. This is a good standard, with growing interest from those doing B2B automation. ebXML provides just enough good technology to make it useful in the real world, without over-hyping its capabilities, thus disappointing its implementers.

What is unique about ebXML is that it's a complete standard, addressing:

- Process
- Trading partner management
- Semantics
- Notation
- Security

- Agreements
- Standard information exchange
- Standard information structure

Other standards, such as Business Process Execution Language for Web Services (BPEL4WS), address only the notion of process and semantics, or other more narrow aspects of application integration between trading partners or internal systems.

However, the aggressiveness of ebXML is also its most limiting factor, because it will take years before the standard finds its way into many enterprises and trading communities. This is due to the amount of work that must be done to get a trading community to leverage ebXML.

ebXML for Trade

The ebXML standard was created to replace EDI (at least, that was the idea) or other electronic commerce standards currently in use. Moreover, ebXML looks to create a set of standards that is affordable and will work in small to medium-sized enterprises. The ebXML standard is open and free to anyone with an interest, and does not seem to compete directly with other standards when you consider its application (but it could have some overlap with other process integration standards, such as WFMC or BPMI).

At its essence, ebXML is based on process models and encoded in XML. It is also an XML message system to exchange information and a repository to allow information sharing. The message system supports any type of data, including EDI transactions and binary information. In addition to information, ebXML supports trading partner agreements – a fundamental function of EDI partner/profile subsystems – and you

may use ebXML to express business services agreements.

As with other standards, ebXML is not a product but a set of guidelines that allow application and application integration technology vendors to design their products to support it. To date, there are dozens of vendors and products that support ebXML; some in part, some completely. More will be added as the standard matures and adoption continues.

In recent years, it's been clear that ebXML (as well as many other modern Internet standards) has to take on a coexistence strategy rather than a replacement strategy. This is because most enterprises are reluctant to shut down their existing B2B systems, such as EDI, until new standards have proven their operational value. Thus, we have another evolution not revolution, which seems to be a common theme as we migrate to newer but more complex and invasive standards.

ebXML Components

There are several components to ebXML, including:

- Collaboration Protocol Profile (CPP)
- Collaboration Protocol Agreement (CPA)
- Business Process and Information Modeling
- Core components
- Messaging
- Registry/Repository

CPP describes an enterprise offering using a standard, portable format. This component describes the message-exchange mechanisms as well as business collaborations that are native to the enterprise or trading community. The ebXML standard also describes business

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.

processes within CPP, including how partners interact within a trading community. CPP supports intra- and inter-company processes, and public versus private processes, collaborating on both sides of a two-party B2B transaction. For example, when leveraging CPP, a trading community would define all processes between partners—for instance, buying parts to build a car—as well as semantic differences, and how processes and data need to interact to support any number of business activities.

CPA describes the particular requirements, facilities, and descriptions for the transaction of trading partner business. It is formed from either manual or automated systems, deriving the intersection of their agreed-upon CPPs. Thus, the CPA becomes the de facto contract between the trading partners, creating “rules of engagement” for a specific collaborative business transaction.

Business Process and Information Modeling is a specification for describing a business process in XML. This includes transactions, document flow, information encryption, binary collaborations, semantics, and such. Processes that leverage ebXML use these specifications when they create CPPs, which are also used to define shared business processes within a trading community.

ebXML and BPEL4WS

You should note that BPEL4WS focuses on the process alone, and not a more holistic view of electronic business. In other words, BPEL4WS is general purpose, whereas ebXML is applicable to B2B trading.

Core components are a set of ebXML schemas and other components that contain formats for business data, including customer account, amounts, and so on. They are particular to an entity, such as a particular trading partner, but not leveraged as vertical semantics (as with vertical standards within health care and financial services markets).

Messaging, as you may expect, is a standard format for ebXML messages that leverages concepts from messaging middleware, including the ability to transmit information asynchronously or synchronously. This is the visible portion of the CPA and provides specific business rules for processing. ebXML messaging is built inside SOAP, extending the SOAP protocol by frameworks that support attachments, security, and verifications of delivery.

The ebXML message service provides a mechanism to exchange business messages that does not rely upon

proprietary technologies and solutions. The message contains structure for message headers used for routing and a payload section for the content.

The ebXML message service is broken down, at least conceptually, into three parts:

1. The abstract service interface
2. Functions provided by the messaging service layer
3. Mapping to the underlying transport service

Registry/Repository are services that maintain CPPs, CPAs, ebXML components, and other ebXML artifacts such as JAR files, video, WSDL, and semantics. This is a database, when you get right down to it, providing query capabilities that allow users to look for relevant content and even information about trading partners. The services defined by CPPs can be published to UDDI.

This portion of ebXML provides a Web service discovery scenario. Thus, a trading partner would first search for a shared service in UDDI, which may indeed contain a reference to a CPP that actually exists in the ebXML registry, which in turn provides access to information about the trading partner. From there you can use the CPA to create a partnership agreement for B2B transactions. We'll talk more about Registry/Repository services in the next section.

The ebXML registry provides a core set of services that enable the exchange of

information between trading partners. As you may recall, the registry is like a database, allowing a trading partner to place and obtain information pertaining to the interaction between trading partners.

The registry maintains an interface to metadata for a registered item, and access to an ebXML registry is gained through the use of several APIs. To facilitate semantic recognition of business process and information meta-models, the registry provides a mechanism for incorporating human-readable descriptions of registry items. Moreover, you can assign UID keys from other existing business processes and information meta models, such as RosettaNet, and implement them using XML syntax.

Conclusion

Now we know what ebXML is, and its power. It's also surprising to me to see how slowly ebXML is finding its way into larger organizations, even those looking to replace EDI with something much more dynamic and process-oriented. It may be another case where the best technology, or standard, does not always win. However, I'm hoping from this column that we learn from the architecture of ebXML, and understand its place in the world of B2B application integration. We have to get off of the EDI train at some point. It's just too slow. ☹

LINTHICUM@ATT.NET

XML-J ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com	978-816-1600	36
Assande	www.assande.com		4
BEA Dev2Dev	http://dev2dev.bea.com/workshop3		2
CTIA Wireless 2004	www.ctiashow.com		9
Java Developer's Journal	www.sys-con.com/jdj	888-303-5282	21
Mindreef	www.mindreef.com	603-465-2204	35
SYS-CON Media	www.sys-con.com	888-303-5282	11
The Gilbane Conference	www.lighthouseseminars.com		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.



Rendering a Connected Tree Using XSLT

The ideas behind XSLT and XPath make perfect sense

One of the things I enjoy most about working with XML and its related technologies is that there is always more to learn. I have been using XML for three years, but not a week goes by that I'm not pleasantly surprised to find something new I can do with it – or, even better, to find an easier way to do something I thought I had already figured out.

One such discovery occurred recently in connection with the rendering of “connected tree”-style hierarchies. A connected tree is one in which lines link nodes to their parents and children, as in many types of file system and outline displays. (See Figure 1 for an example.) The application we are currently building at PortBlue uses this type of display in many places for different purposes, making it difficult to unify handling of the rendering process across the varying types of data in the hierarchies. However, all the hierarchy types are available as DOM trees. It struck me that if I could find a way to render connected trees using only XSLT, I could remove all the application code associated with tree rendering and use a single stylesheet to handle all the tree rendering, which would vastly simplify our code.

Implementing this idea has resulted in several hundred lines of bug-prone Java rendering code (spread across many classes) being replaced with a few dozen lines of XSL in a single stylesheet, improving our system's reliability, maintainability, and ease of customization. In this article, I'll explain how we did it.

As I'm sure most readers are already aware, XSL and XSLT are paired standards that specify how to describe a transformation between XML and some other desired format for the same data.

An XSL stylesheet provides a recipe for finding data in the source XML data, transforming it, and outputting the transformed version. The technology has a multitude of uses, but probably the most commonly encountered application is the transformation of XML data into appropriate HTML for presentation to a human user.

Yet another specification, called XPath, is used extensively in XSLT. XPath is a functional language for locating information in an XML tree structure. Its role in XSLT is as a structural pattern-matching tool for finding the data that is to be transformed into particular parts of the generated document.

For me, at least, XPath has been the hardest part of the XSLT toolkit to learn. One aspect of XPath that gave me particular trouble is the use of “axes” to send the XSLT processor through the document in different “directions.” The technique I will describe here depends on three different axes being used, and hopefully will provide a useful introduction to the power of axis specification for those joining me in climbing this learning curve. But before going on, let's examine what is involved in rendering a tree like the one shown in Figure 1.

Four Sticks Make a Tree

If you examine Figure 1 closely, you'll notice that there are only four graphic shapes needed to draw the tree. First, each element is connected to the tree by a graphic with a horizontal prong going from center to right. If it is not the last child element of its parent, the prong is attached to a bar that goes all the way from top to bottom, giving it the shape of a capital T rotated 90° left. The last child is missing the bottom half of the vertical bar, giving it the shape of a cap-

ital L, or of the lower-left corner of a square.

If the element is more than one level deep in the hierarchy, there will be other tree graphics to the left of the connector graphics just discussed. For each level of “ancestor” of the current element, one of two graphics appears. If that ancestor is not the last child element of its own parent, then a vertical bar is shown, part of a series of such lines leading down to the next sibling element at that level. If that ancestor is the last child of its own parent, we instead display an empty spacer graphic.

For purposes of this article, I have created four images, *tree_tee*, *tree_corner*, *tree_bar*, and *tree_spacer* (see Figures 2; images are also available at www.sys-con.com/xml/sourceec.cfm). Each meets the descriptions given here, with the exception that I've added a visible dot to the center of the spacer image to make it clear where it is being used.

With the above analysis in place, we can write an informal algorithm for how each line of a tree is to be rendered:

1. For each level of ancestor from the top of the tree to the parent of the current element, draw a spacer if that element is the last sibling under its parent, a bar if it is not.
2. Draw a corner if the current element is the last sibling of its parent element, a tee if it is not.
3. Draw the identifying information for the current element.

One of my coworkers eloquently dubbed the line of graphics generated in steps 1 and 2 the “stick stack,” which is actually a pretty good way to think about it; you're building a stack of sticks (connector graphics), one for each level from the root to the current element. The

AUTHOR BIO

Craig Berry is principal architect at PortBlue Corporation (www.portblue.com), a software company in Los Angeles that offers an advanced expert system platform built on J2EE technology. He has 20 years of experience in software design and implementation in fields ranging from AI research to distributed streaming media. He holds a BS degree from Harvey Mudd College.

rightmost graphic will always be a tee or a corner, and those further to the left (if any) will always be bars or spacers. Similarly, we use “item display” to cover what you render to the right of the “stick stack” to provide information about each element – in the case of our example, this is simply the element’s title, formatted according to the element type. So you can further summarize the process as “For each element in the tree, draw the stick stack, then the item display.”

As usual, the devil is in the details. So let’s turn our attention to how this is implemented in XSLT.

Under the Hood

The source document used to produce Figure 1 is called “book.xml” and appears as Listing 1. It is a very simple XML document containing the outline of a book. The top-level book element contains chapter elements, and these contain section elements, which may themselves contain other sections. Each element has a title attribute, which we will use to display its name in the tree. Obviously, this is a very simplistic document, but the principles we will be using to transform it apply equally well to more complex documents.

The stylesheet that we will be applying to the source document is called “connected_tree.xsl” and appears as Listing 2. Again, this is an artificially simple stylesheet, but the concepts it illustrates can be extended to more ambitious transformations.

Lines 1–13 of the XSL file are boilerplate, standard header material that would appear at the top of any XSL file being used to map into HTML.

The more interesting stuff starts on line 17, with the definition of a template that will match the top-level book element in the XML document. Within this template, we generate the static HTML components for the output page. One of these is an embedded CSS style definition section in the header, which we use to adjust margins and borders so that our tree graphics will join pixel-to-pixel without unsightly gaps, and also to adjust vertical alignment so text will flow cleanly from the ends of the rightmost connector graphics.

On line 28 we output the title of the book (that is, the value of the title attribute of the book element), enclosed in `...` tags so that it will be displayed in boldface. Then, on line 29, we apply relevant templates to the children of the book element, operating in “line” mode. In this and other cases of applying templates to children, the child elements

will be processed in document order. As each will in turn do its own processing and then apply templates to its own children, the desired “tree-order” output generation is accomplished.

XSL modes allow us to process the same element in different ways in different circumstances, which you’re about to see is the key to this solution. “Line” mode selects templates that handle rendering of the full output line, including the stick stack and the item display. The other modes used here are “item” and “tree,” which we’ll discuss later.

All element types handle “line” mode the same way, so there’s only one “line” mode template, matching all element types; it begins on line 36. Within this template we will generate all of the output for one horizontal line of the tree diagram. We wrap the display in a `<div>` element of the appropriate CSS class to achieve the desired zero-border, vertically centered display. Within the `<div>` we find two predictable components. The first, on line 38, calls the “graft” template (as in “graft this branch onto the tree”) to build the stick stack; we’ll discuss how that works below. The second, on line 39, applies templates to the current node again, but this time in “item” mode rather than in “line” mode. Outside the `<div>`, at the end of the template, we do all this in turn to the children of the current element, continuing the full-tree traversal.

“Item” mode is used to achieve the item display rendering mentioned previously. Each element type will typically do this differently. For this example, each of the two element types that can appear in a book (chapter and section) has its own “item”-mode template (see lines 46–52). Again, we simply output element title attributes with different formatting for each type.

And with this, at last we reach the fun part. I mentioned above that the “graft” template (which starts on line 57) builds the stick stack for the element on which it is called. How does it do this? The easiest way to approach this question turns out to be backwards, conceptually from right to left on the line. So skip down to line 62, where we generate the connector graphic – the one that will be right next to our item display, with a “prong” sticking out to the right. As we noted above, this will be a corner graphic if this element is the last child of its parent, a tee otherwise. If you read the source, that’s exactly what it says – the tricky part is understanding how it determines if the current element is the last child of its parent.

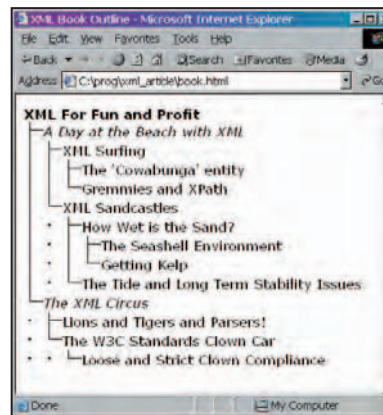


Figure 1 • Rendered tree

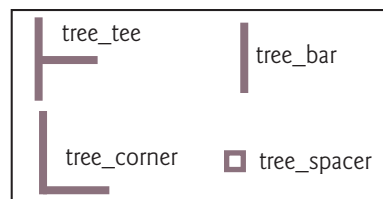


Figure 2 • Tree components

The key to accomplishing this is the use of an alternative “XPath axis.” For most common purposes, the default *child* axis is what you want; it selects the children of the current node. We’ve used it extensively to get this far. However, there are many other axes you can use to traverse the source document in various “directions.” The first one we will discuss is *following-sibling*, which (not surprisingly) selects the siblings (children of the same parent) of the current element that are later in document order. You can see it being used on line 63, in a conditional test. If “following-sibling:*” returns a non-empty node set, it is considered true for test purposes and we choose the tee graphic; otherwise, if the set of following siblings is empty, this counts as a false result, and we display the corner graphic. The *following-sibling* axis (and its complement, the *preceding-sibling* axis) are extremely useful in any case where you need to do something special at the ends of a list of sibling elements.

So that’s how the rightmost stick in the stack is rendered. But what about the rest of it? As you’ll recall, there should be one graphic for every chapter or section that is an ancestor of the current element, with each being a bar or spacer depending on whether that ancestral element has following siblings. Half the solution should be obvious from how I phrased that, but how do we visit our ancestors to make the decision?

This time, the trick is another XPath axis: *ancestor*. On line 59, you’ll notice that we are applying templates in “tree”

mode to a node set selected as "ancestor::*". This expression means "all the ancestors of the current node, from the root downward, excluding the node itself." This is exactly the set for which we need to generate "sticks," which occurs in the two final templates, each tagged as being applicable to "tree" mode.

The first, on line 74, simply suppresses generating a graphic for the root (book) element. We're dealing with that element separately, in the "/book" template we discussed earlier, so we want to ignore it for tree-rendering purposes.

The second, starting on line 78, uses logic just like that we discussed above to choose between a bar or a spacer graphic based on the presence or

absence of following siblings.

The Road (to XSLT) Goes Ever On and On

And that's it! If you apply the stylesheet to the XML source, you will obtain HTML that produces a display like that shown in Figure 1. There are numerous ways to do this; my preferred technique for XSL experimentation is to use the Apache Xalan command-line mode, which (assuming you have it installed, working, and in your CLASSPATH) looks like this:

```
java org.apache.xalan.xslt.Process
-IN book.xml -XSLconnected_tree.xsl
-OUT book.html
```

Note that this article examines just one way to do the connected-tree rendering trick, and there are endless ways to customize, enhance, or otherwise fiddle with it. In my own experience, the key to learning XSLT has been experimentation, so I encourage you to try changing both `connected_tree.xsl` and `book.xml` in various ways to see what happens. A lot of the ideas behind XSLT and XPath can be counterintuitive at first, but gradually you'll discover that it all actually makes a rather beautiful kind of sense. And then you'll be able to share your own interesting tricks with the world.

CRAIG.BERRY@PORTBLUE.COM

LISTING 1 •

```
<?xml version="1.0"?>
<book title="XML For Fun and Profit">
  <chapter title="A Day at the Beach with XML">
    <section title="XML Surfing">
      <section title="The 'Cowabunga' entity"/>
      <section title="Gremmies and XPath"/>
    </section>
    <section title="XML Sandcastles">
      <section title="How Wet is the Sand?">
        <section title="The Seashell Environment"/>
        <section title="Getting Kelp"/>
      </section>
      <section title="The Tide and Long Term
        Stability
        Issues"/>
      </section>
    </chapter>
    <chapter title="The XML Circus">
      <section title="Lions and Tigers and Parsers!"/>
      <section title="The W3C Standards Clown Car">
        <section title="Loose and Strict Clown
          Compliance"/>
        </section>
      </chapter>
    </book>
```

LISTING 2 •

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output
    method="html"
    encoding="UTF-8"
    indent="yes"
    version="1.0"
    omit-xml-declaration="yes"
    media-type="text/html"
    standalone="yes"
  />

  <!-- Templates used to generate text content -->

  <xsl:template match="/book">
    <html>
      <head>
        <title>XML Book Outline</title>
        <style type="text/css">
          body { font-size: smaller }
          div, img { border: 0px; margin: 0px; padding: 0px }
          div.Node * { vertical-align: middle }
        </style>
      </head>
      <body>
        <b><xsl:value-of select="@title"/></b>
        <xsl:apply-templates mode="line"/>
      </body>
```

```
</html>
</xsl:template>

<!-- Show each tree line -->

<xsl:template match="*" mode="line">
  <div class="Node">
    <xsl:call-template name="graft"/>
    <xsl:apply-templates select="." mode="item"/>
  </div>
  <xsl:apply-templates mode="line"/>
</xsl:template>

<!-- Show item displays for different element types -->

<xsl:template match="chapter" mode="item">
  <i><xsl:value-of select="@title"/></i>
</xsl:template>

<xsl:template match="section" mode="item">
  <xsl:value-of select="@title"/>
</xsl:template>

<!-- Templates used to generate the "stick stack" of
  tree connectors -->

<xsl:template name="graft">
  <!-- Generate ancestor connectors -->
  <xsl:apply-templates select="ancestor::*" mode="tree"/>

  <!-- Generate current-node connector -->
  <xsl:choose>
    <xsl:when test="following-sibling::*">
      
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!-- Suppress ancestor connector for the top node -->

<xsl:template match="book" mode="tree"/>

<!-- Show ancestor connectors for all other node types -->

<xsl:template match="*" mode="tree">
  <xsl:choose>
    <xsl:when test="following-sibling::*">
      
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

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!”

Try It online at **XMETHODS.net** or download a trial at Mindreef.com



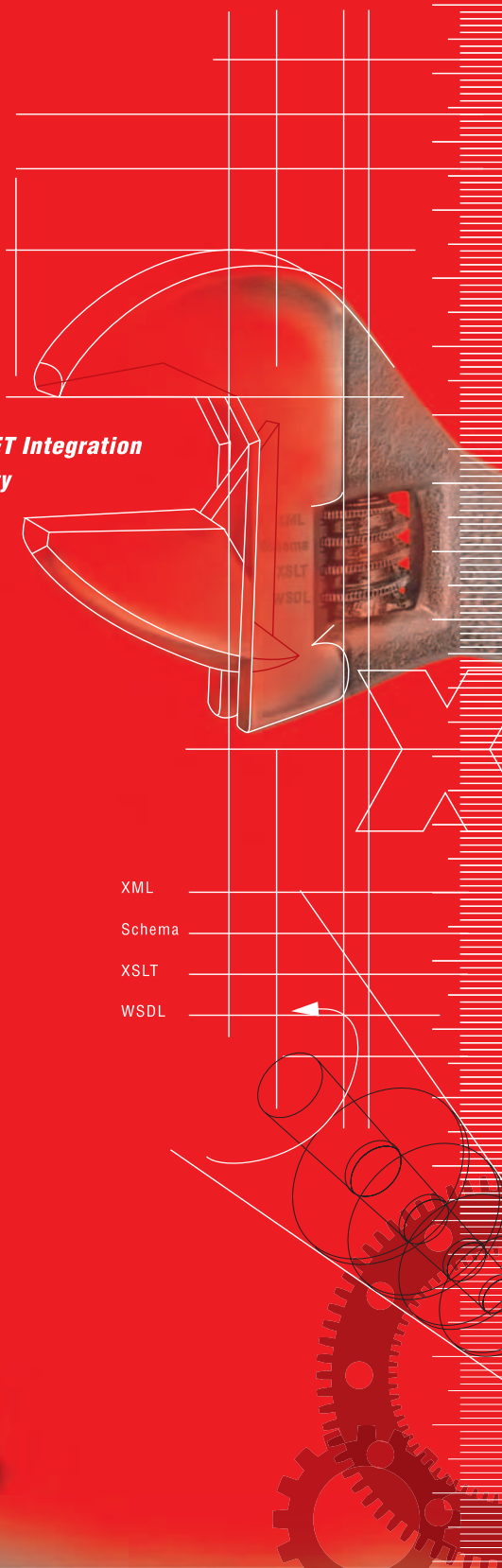
© Copyright 2004, Mindreef, Inc. The names of companies and products mentioned herein may be the trademarks of their respective owners. This product uses Hypersonic SQL. This product includes software developed by the Politecnico di Torino and its contributors.

“The Web Services Diagnostic Experts” **Mindreef**



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

ALTOVA®

www.altova.com