Content Management • Data Management • XML Labs

XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

January 2003 Volume: 4 Issue: 1

xml-journal.com

$6.99US $7.99CAN
0 09281 01314 3 02

SYS-CON MEDIA

XML JOURNAL

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 sim@polarisventures.com

EDITORIAL
Editor-in-Chief
Hitesh Seth hitesh@sys-con.com
Editorial Director
Jeremy Geelan jeremy@sys-con.com
Managing Editor
Jennifer Stilley 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

PRODUCTION
Production Consultant
Jim Morgan jim@sys-con.com
Art Director
Alex Botero alex@sys-con.com
Associate Art Directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Assistant Art Director
Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE
James Brannan, Chris Halaschek, Michael Hanson, G. Ken Holman, Ayesha Malik, John A. Miller, Vladimir Rasin, Richard Salz, Hitesh Seth, Aaron Skonnard

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.

SYS-CON MEDIA

The Road So Far... and the Road Ahead

WRITTEN BY **HITESH SETH**

One of the key reasons behind the explosive growth of the Web is the simplicity and ease of use of the underlying standards – TCP/IP, HTTP, and HTML. Experts and critics have often argued about what's wrong with HTML and HTTP and why we need IPv6 et al. For instance, with the advent of XHTML, we are now realizing why HTML should have probably been XML-ized from Day 1; similarly, we criticize the fact that HTTP is stateless. However, the ubiquity of these standards has easily surpassed their technological imperfections.

Enter XML, and the definition of ubiquity changes altogether; what started as a simple markup language for information dissemination is now generalized as a generic application and service delivery platform for any kind of device.

Where have we seen the effectiveness of XML, and in what areas do we feel that XML has a way to go, either in adoption or in inception? Areas where XML has been effective include:

- ***Core DOM/SAX APIs for XML:*** These have undergone significant release cycles and have become quite stable for processing and generation of XML.
- ***Data transformation and processing:*** XSLT has already replaced a number of proprietary mapping interfaces (watch for XML Query here).
- ***Wireless applications:*** We saw an initial growth of standards, particularly WAP, that used XML but deviated from the world of HTML-based Web applications. We all believed that we had to have two separate presentation markups: HTML for Web and WAP for wireless (our only savior being XSLT). Reason intervened and we got XHTML into the making, and we realized that wireless/Web application development should use XHTML Basic as the core markup language.
- ***Core Web services:*** Even though analysts believe that Web services isn't yet prime time, we've seen a rise in use of core infrastructure Web services standards such as SOAP and WSDL. However, I believe we're still far from reaching the automated nirvana that UDDI-based service lookup and binding will provide (watch for standardization of Web services business process orchestration here).
- ***Industry vocabularies:*** Some argue that the growth in the number of industry vocabularies using XML has become too explosive. This may be true, but it's been shown that where serious groups of companies have collaborated, standards have materialized and become critical to that industry. Take CIDX (Chem XML) for instance. It has created a definitive space for itself in the chemical industry. Where we have lacked is in areas that require cross-industry definition of business process templates and documents (watch for developments around ebXML, OAGIS), where EDI is strong with a huge user base.

Some of the areas in which I believe XML still has journey ahead include:

- ***XML Schemas:*** I mention XML Schemas here because even though XML Schemas have been available as a W3C Recommendation for some time, we have yet to see widespread development around them. In my consulting with technology clients, I see that a number of developers have still a way to go before they migrate their loose DTDs into strict schemas. This is probably related to the complexity of the schema specifications and also the base education required around them.
- ***Speech and/or multimodal applications:*** We've seen initial developments around VoiceXML, SALT, XHTML+Voice, and CCXML, and I believe there is consolidation and further standards development that can standardize the way we develop interactive, natural language–based speech applications.
- ***XForms, XHTML:*** We love XML's ease of use but hate the loose HTML code that is floating around in the majority of Web sites and applications. XHTML goes one step further to XML-ize the basic HTML code itself, requiring significant changes to the existing code and coding style. The new XForms specification is an interesting addition to this mix, providing a strong next-generation forms solution for device-independent Web applications.

Although we've faced some challenges (for example, the development of XML Schemas), the road so far has been quite smooth for XML. Looking at some of the key issues that we'll have to tackle going forward – XML Query, common business documents semantics, Web services orchestration, and so on – we're likely to hit some bumps in the middle.

As the new editor-in-chief of *XML-Journal*, I plan a strong focus on technology and real-world applications, a combination that I believe can be very effective in navigating XML's exciting journey.

AUTHOR BIO

Hitesh Seth, editor-in-chief of XML-Journal, is the chief technology officer of ikigo, Inc., a provider of XML and Web services monitoring and management software.

HITESH@SYS-CON.COM

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9637

PRESIDENT and CEO
Fuat A. Kircaali fuat@sys-con.com

BUSINESS DEVELOPMENT
VP, Business Development
Grisha Davida grisha@sys-con.com

COO/CFO
Mark Harabedian mark@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
Advertising Account Manager
Megan Ring-Mussa megan@sys-con.com
Associate Sales Managers
Carrie Gebert carrieg@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Alisa Catalano alisa@sys-con.com

SYS-CON EVENTS
President
Grisha Davida grisha@sys-con.com
Conference Manager
Michael Lynch mike@sys-con.com
Regional Sales Managers, Exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richarda@sys-con.com

CUSTOMER RELATIONS
Customer Service Representative
Margie Downs margie@sys-con.com

JDJ STORE
Manager
Rachel McGouran rachel@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
Kerri Von Achen kerri@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

Web Service Abstractions

WRITTEN BY **AARON SKONNARD**

Joel Spolsky, a long-time blogger who frequently publishes interesting software development observations, recently wrote a seminal piece entitled "The Law of Leaky Abstractions"(www.joelonsoftware.com/articles/LeakyAbstractions.html). I encourage you to take a few minutes to read it, especially if you're involved with XML Web services. It summarizes one of the growing and troublesome trends surrounding Web services development and tools.

Joel's essay concludes that all nontrivial abstractions leak to some degree – they don't fully hide the underlying technologies they were originally intended to encapsulate and simplify. For example, most people drive cars, yet only a small percentage of the population understands anything about what goes on under the hood. This is a wonderfully successful abstraction that allows countless automotive ignoramuses (like me) to zip around the countryside at impressive speeds.

However, if you're driving and the dashboard lights start flashing and smoke begins to emerge from under the hood, you have to know about the car's underlying technologies to address the problem. Most people would at least stop the car, open the hood, investigate a bit, maybe check the oil, let the car cool down, and see where that leaves them.

Things go wrong all the time and when they do, abstractions leak. In this case, the car manufacturer isn't able to completely hide the inner workings of the car when it doesn't work as intended. In fact, to fix the problem someone who knows a great deal about automotive internals (the mechanic) will undoubtedly get involved. There are countless examples of abstraction leakage, especially in the realm of software development.

The troubling trend is that software developers, by and large, seem unwilling to learn the core XML technologies that lay the foundation for the emerging Web services platform. In fact, most developers seem convinced they can rely on Web service abstractions and develop sophisticated applications without knowing anything about what goes on under the hood, which includes such technologies as XML 1.0, Namespaces, XML Schema, SOAP, and WSDL.

Web service tools vendors aren't helping the situation either. The first-generation toolkits actually support, perhaps even encourage, this dependent attitude. Microsoft's .NET Framework makes it possible to implement Web services as methods on a class:

```
public class MathService {
  [WebMethod]
  public double Add(double x, double y) {
    return x+y;
}
```

This powerful abstraction hides many Web service complexities by allowing developers to work with a more familiar object paradigm. Developers should embrace this abstraction if it helps increase their productivity. The problem occurs, however, when developers believe the abstraction to be airtight, that they'll never need to understand things like XML.

Imagine a Java developer who wants to consume this simple .NET Web service with Apache's Axis toolkit as shown here:

```
MathService math = new MathService();
double sum = math.Add(33, 44);
```

The abstraction seems tight, but what if Add fails? It doesn't take much for Web service abstractions to begin leaking; once they start, they quickly begin to gush. The Add invocation could fail for a variety of reasons. It could encounter a networking problem like a DNS resolution error, a TCP timeout, or an HTTP authentication failure. It could also fail due to an incorrect XML 1.0 message for the given operation. For example, the namespace might be off by one character. Such a problem may have been derived from a bad XML Schema definition or a bug in the WSDL document. Ultimately, to get to the bottom of the problem, the Java developer will have to know something about the XML platform.

As Joel so eloquently wrote, "abstractions save us time working, but they don't necessarily save us time learning." Building Web services is just like driving a car, but when things go wrong, remember that we're the mechanics. Nothing will empower you more in building Web services than embracing the XML platform and its core technologies, namely XML 1.0, Namespaces, XML Schema, SOAP, and WSDL. Doing so will help you maximize the return offered by today's powerful but often leaky Web services abstractions.

AUTHOR BIO

Aaron Skonnard is an instructor and course author at DevelopMentor, where he focuses on the XML and Web services curriculum. Aaron coauthored Essential XML Quick Reference *and* Essential XML *(both from Addison-Wesley) and currently writes The XML Files for* MSDN Magazine.

AARONS@DEVELOP.COM

Reader Feedback

Stirring the Pot: XML Enablement

Viewed 50,000 times by *XML-J* readers around the world, *XML-J*'s news item covering Office 11 sparked heated discussion in the days following its original publication. Here's a sampling of what readers posting to the *XML-J* Web site had to say…

So what happens?

So, what happens when somone wants to e-mail an XML-enabled Word document.... Does it somehow become encrypted on its way out of the database, remain scrambled on its way over the Internet, and reassemble itself into nice XML once it arrives on the recipient's computer? Doesn't sound like XML to me?!
-Tony Weston

Okay, so it'll be harder

Okay, so it'll be harder to mount a Windows partition effectively, but this doesn't affect transmission of documents, especially if they're stored in an XML format. As for me, I think it's more valuable to have files that I can read outside of their native filesystem than to have a readable filesystem filled with unreadable files.

It will indeed be harder

It will indeed be harder to mount the partition. It may also be harder to use that XML data, since what we may be talking about is XML encapsulation of binary, proprietary, encrypted file formats. Don't necessarily think you're going to receive at the other end a plaintext file with a few tags – what you will receive will have been through a closed kernel "request" to an encrypted database "filesystem," a proprietary DRM system (hardware and software) – and you genuinely believe they're just going to bang it out as plaintext at the other end?

MS routinely uses XML to encapsulate

MS routinely uses XML to encapsulate (proprietary) binary data. In the case of the MS Office file format, this is especially true, but to a lesser extent this also goes for stuff like BizTalk, etc. (which has a terrible license attached to it). If MS is *really* serious about using open formats, and using XML in their Office suite, they should put their money where their mouth is and join in the OpenOffice File Format project [openoffice.org]. Most of the open-source players are working there already, and the EU is also set to join. Mature participation of Microsoft would be very welcome. Of course, this will never happen. Instead, MS will continue to push their own "open" XML-based file formats. Microsoft Kerberos, anyone?

Office or plaintext

Think of it: when you exchange files with other businesses, you have two realistic choices of file formats: Office or plaintext. I think PDF is a viable (even growing) third option. Adobe is "evil" just like MS (remember Sklyarov). Regardless, PDF is nice and works well, and the files are way smaller than Word docs.

Too good to be true

Maybe they need a migration path away from the win32-based format they use now. .NET also seems to follow that path. Remember that MS needs access to platforms other than the i386/desktop in the future – mobile devices, for instance. Keeping a format that is basically a binary image from a PC is good for locking out competition, but not when you have to start competing with yourself.

What will be the default save format?

The most important question, besides whether the MS Word XML format will be well-documented enough, is whether it will be the default saving format. Most MS Office users simply don't care enough to save MS Word documents in RTF, for example, even if it's more than good enough for the vast majority of the documents. Not the main issue of the article, but it is unfair to single out someone as the inventor of XML, which is just a streamlined version of SGML, which is an evolution of IBM's GML.

"If MS is really serious about using open formats, and using XML in their Office suite, they should put their money where their mouth is and join in the OpenOffice File Format project"

Historical turning point?

I just thought about someone saying that somewhere, when you look back in history, you can see a turning point where things just went wrong or right. One small such point is when IBM gave out the specs to their PC hardware, allowing everyone to clone it, while Apple did not. This may be another of those times. Maybe in 10 years we'll look back at this and ask ourselves "Why the heck did MS XML-enable their Office app, releasing the hold that they had?" Only time will tell, I guess.

Codename?

It is interesting that "Office 11" is referred to as a codename for the next version of Office. Really, it is simply the version number. Office XP is Office 10 (start any Office app, click on Help -> About and look for yourself). To call it a codename is mystifying it.

One correction

Replying to vidarh: "The point of XML is not for it to be human readable, but to allow easy automatic processing of various kinds"...
XML spec, section 1.1, Origin and Goals:

- Goal 4: "It shall be easy to write programs which process XML documents."
- Goal 6: "XML documents should be human-legible and reasonably clear."

The point of XML is /*both*/ of these things.

Tim (Bray) here with a bit more background

I've seen the native Word XML format (alpha mind you, so it might get changed). It isn't exactly pretty, and if I had to write code to extract all the paragraphs that contained the word "foo" in bold it would give me a bit of a headache, but I could do it.

The word "foo" in bold single-underline looks something like : **foo**

Yeah, it's pretty verbose.

Near as I can tell, it is 100% round-trip-able, i.e., you save as that file format, you read it in again, you hit ctl-S and it saves again; about as good as a native format. Now someone needs to write some scriptware to run Word in batch mode to XML-ify server directories with zillions of office docs.

I think the reason MS is doing this is obvious. Look at their financials – they **really** need people to upgrade to the new version of Office. End users don't buy Office any more; CIOs and the like do. These people are just not going to be impressed by another new word-processing feature, but they might be motivated to upgrade if they thought that they were opening up all their data to reuse by other programs.

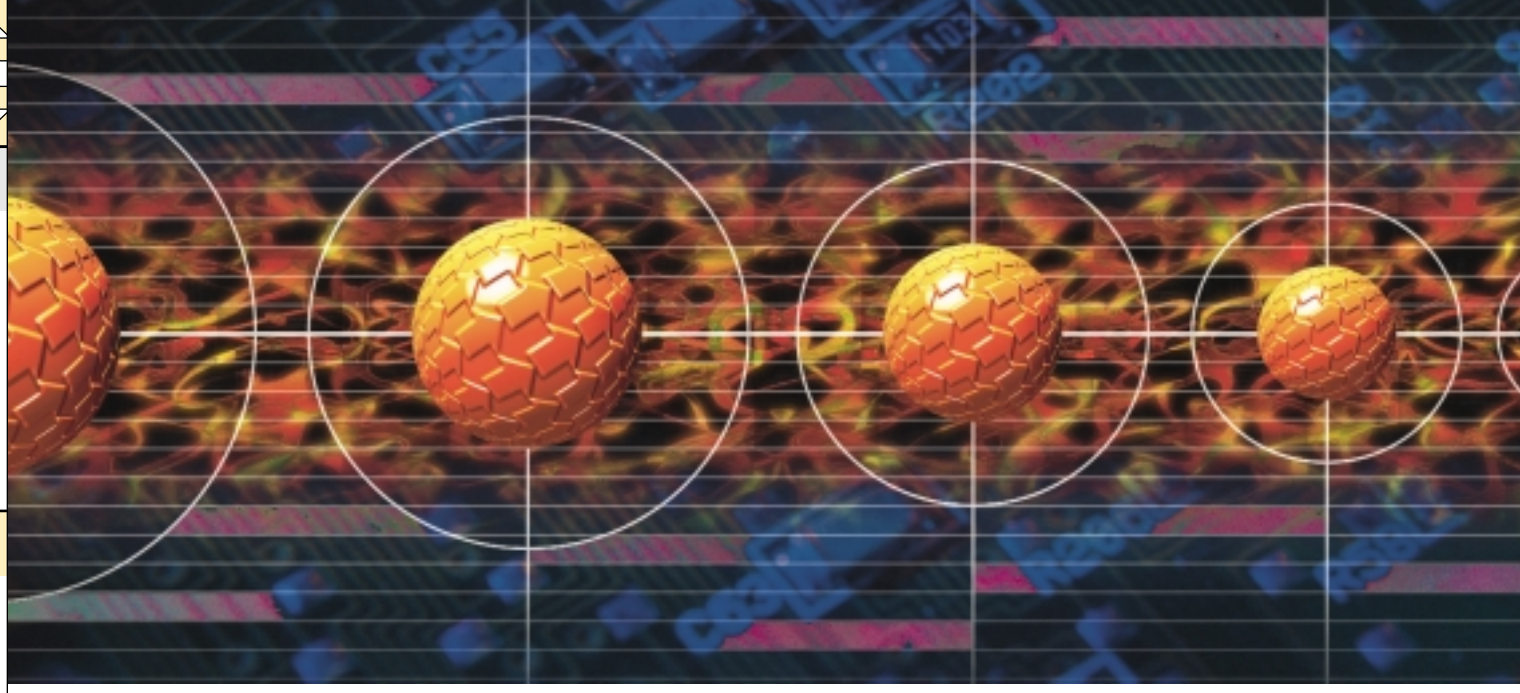
I expect that with any luck we'll get a secondary industry built around doing cool unexpected stuff to Office docs. Don't want to sound over-excited here, but a huge amount of all the intellectual capital in the world is sitting around in Office docs, and this makes it noticeably more reusable. Has to be a good thing.
-Tim Bray

Other Office features

Does Office 11 do anything other than support XML?
-Colin Banfield

Letters may be edited for grammar and clarity as well as length. Please e-mail any comments to Hitesh Seth (hitesh@sys-con.com)

HITESH@SYS-CON.COM



Service Firewalls to Protect Web Services

Using today's technology

WRITTEN BY
MICHAEL HANSON

Walk into a roomful of technology vendors discussing Web services security, and you're likely to choke on all the smoke being blown around. Let's clear the air a little. There is no reason why IT organizations cannot deploy Web services in a secure and manageable way using today's technology.

You don't have to be left behind. This article will outline what it takes to safely expose your business systems over the Internet and keep your applications running safely and cost efficiently.

While Web services standards – UDDI, WSDL, and SOAP – do not directly address security, Web services are based on transport mechanisms that have their own existing security standards. Commercial solutions are available today for architects and developers looking to securely deploy Web services in an interoperable manner over the public networks.

However, when deploying Web services, you must carefully consider what it takes to maintain security for a Web services–enabled application. Once auditors or compliance managers have determined it is safe to deploy an application, partner identities must be managed; certificates must be issued, maintained, and revoked; and transactions must be audited. Additionally, some partners can be trusted to invoke certain interfaces while others cannot. Through all this, developers and administrators struggle to keep up with evolving standards. Keeping these challenges in mind will help you differentiate among Web services security vendors.

The Current State of Web Services Security

While it is possible to secure Web services today – and more is being done to make it easier tomorrow – Web services platforms alone are insufficient to securely deploy Web services–enabled applications. While Web services can take advantage of existing technologies for authentication and authorization, complete Web service security is about more than just access control. Secure Web services deployments must not only implement authentication and authorization capabilities, but also provide content validation, transport- and message-level encryption, digital signatures, a robust logging system, and the ability to effectively manage security to respond to ever-changing business needs.

Application developers who want to implement enterprise-class security for their projects will need more than what the Web Services Security (WSS) specification, currently being developed in OASIS, provides. Web services can take advantage of existing technologies for authentication and authorization, including using bilateral certificates over SSL and exchanging user names in HTTP headers, and these services are currently available in most SOAP activation frameworks. WSS extends these capabilities and enables new security capabilities within the SOAP envelope, but covers only part of the total solution.

The Service Firewall Completes Security

A service firewall is a network element, either logical or physical, that links an enterprise's Web services to external partners or internal consumers in order to secure the bidirectional flow of XML messages. This functionality is separate from what a network firewall provides, and the two should be used together. The service firewall is in the logical data path between organizations. In this position, the service firewall can monitor, secure, and control all Web services traffic.

Shifting this responsibility from the Web services platforms

to a single point allows centralized control over which clients have access to what services, and offers the ability to enforce other global security policies across a distributed application. This concentration of security policy enforcement allows an enterprise to shift to new and emerging security standards quickly and easily without having to make changes to each individual application.

The ability to maintain consistent security policy across multiple applications deployed on multiple platforms is another reason why centralization of security is important. In a fully realized service firewall implementation, each Web service invocation that arrives at an application can be assumed to have been decrypted, authenticated, authorized, validated, and logged, allowing the application to process the request without having to perform each of those actions itself. In addition, centralized security management allows new applications and new partners to be quickly, securely, and inexpensively added. This scheme significantly reduces the complexity of Web services applications and improves the security of the system as a whole.

Service Firewall Capabilities

Service firewalls should provide all of the security services necessary to completely secure enterprise Web services. The WSS specification defines many of the standards that a service firewall must implement, but a service firewall must implement a larger set of features to properly secure an application.

Authentication

Service firewalls can use a variety of means to establish the identity of the client. HTTP/S provides two methods: HTTP Basic-Auth headers and SSL Client Certificates. Both of these methods are applicable to Web services requests. WSS also defines four new means of establishing identity within the SOAP envelope and independently of the transport: user names and passwords, X.509 certificates, Kerberos tokens, and SAML assertions. Listing 1 shows the use of an X.509 certificate to show proof of possession.

Authorization

Once the identity of the Web services client is established, the service firewall must check the entitlements granted to that identity. This can be done by accessing a variety of entitlement repositories, or by having an integrated store for managing entitlement. In the former case, the source could be an enterprise LDAP server, Active Directory, or another database. In addition, certificates can be checked with certificate authorities to determine if they are current or have been placed on a certificate revocation list (CRL).

Encryption/privacy

Encryption is an important part of maintaining message privacy. Web services can use HTTP over SSL to provide point-to-point encryption of messages, and a service firewall should be able to perform all forms of SSL and TLS functions. Depending on the privacy requirements, it may be necessary to use a long (128-bit) RSA key, in which case the help of a hardware accelerator might be appropriate.

WSS defines a method of using XML-Encryption to encrypt portions of a SOAP message and include additional information about how the encryption was performed in the Security header. Listing 2 shows the use of XML-Encryption in a SOAP message requesting credit card authorization.

Signatures

XML digital signatures allow senders of SOAP messages to sign the content of messages before sending them so that receivers of these messages can verify that the contents have not been changed in transit. WSS specifies a means to include signatures in the Security header of a SOAP message. Service firewalls can perform signing and verification functions for both incoming and outgoing messages.

Content validation

Some might assume that an authenticated and authorized request whose contents have been decrypted and whose signatures have been validated should be approved. However, this would ignore a critical aspect of application security, namely, validating the correctness of the request for both semantics and syntax. Checking both is important because no single party controls both ends of the communications channel; therefore, no message, even one that is from a trusted identity, can be assumed to be correct and not from a malicious individual within the trusted partner organization. Service firewalls need access to a repository of DTD and XML Schema files to perform this content validation.

Logging

Service firewalls should be able to log each message in a reliable and auditable way and report on what SOAP messages were sent by whom, when, with what parameters, and with what result. The best practices regarding log file retention suggest that log files may need to be kept for as long as seven years depending on their role in business processes (such as for accounting and billing). This is important, and when considering service firewall architectures, the logging system must have archival capabilities in order to fulfill its

Glossary of Terms

WSS
The OASIS technical committee known as the Web Services Security TC is working on standardizing and furthering the work originally done by IBM, Verisign, and Microsoft on a foundation of security services for Web services. www.oasis-open.org/committees/wss

SAML
A standard for exchanging XML-based authentication and authorization information. The specification can be applied to such varied applications as multidomain single sign on and SOAP messages. www.oasis-open.org/committees/security

XML-Encryption
Governs the process of encrypting and decrypting content, providing an XML syntax that represents both the encrypted data and pointers to help the decrypting party. It can be used to encrypt entire XML documents, or to encrypt only specific elements within XML documents. www.w3.org/Encryption/2001

XML-Signature
Defines a syntax for encapsulating or embedding a digital signature for an XML or other digital document. The specification allows senders to attach their identity to content, and allows receivers to verify that the content has not been altered in transmission. www.w3.org/Signature

XKMS
Specifies a method for simple clients to obtain key information from a Web service, or to publish key information to a repository. The specification allows easier distribution, registration, and verification of X.509 certificates. www.w3.org/2001/XKMS

ADOS Co., Ltd.

http://www.a-dos.com

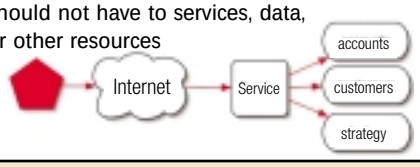
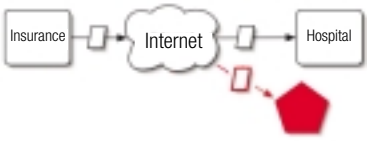
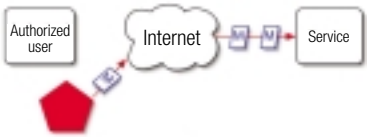
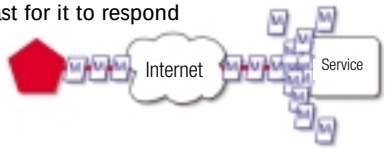
role as a central log for all inter-enterprise Web services traffic. Because of this, the service firewall should be deployed on an open hardware platform that allows administrators to configure hardware to satisfy logging requirements, such as a large disk array or a tape backup unit, both at the time of deployment and as part of the ongoing task of maintaining a service firewall.

Management

All of the security functionality mentioned above could be implemented as a stand-alone service firewall or integrated into a Web services application. The stand-alone service firewall, however, can be managed much more easily and at a lower cost than a custom application; when providing security for more than one application, these advantages become

Understanding Security Threats

Security experts classify network attacks according to the kind of threat each poses. This classification is called the threat model of an attack. An attack may have more than one threat model at a time, and schemes for classifying threat models vary; the important thing is that a representative threat model describes the main ways that a network can be attacked, and the countermeasures that can be used to defend it.

Threat	Description	Countermeasure
Authentication	Attacker pretends to be an authorized user 	Cryptographic technology such as SSL to establish a secure connection between the user and the service, and then digital certificates exchange to ensure that each party is who it claims to be
Authorization	An authenticated user obtains access that he should not have to services, data, or other resources 	Access to different services must be tightly controlled and managed
Confidentiality	Attacker can gain access to sensitive data 	Cryptographic tools provide the most effective protection against loss of confidentiality, enabling networks to transmit sensitive data in an encoded form that is useless to attackers
Data Integrity	Attacks that rely on altered or malicious data 	Techniques based on digital signatures can guarantee that a message cannot be altered without the receiver knowing about it
Replay	Attacker repetitively resends an intercepted message 	Attach a serial number or identifier to each message and compare each new incoming message to ensure that no message is used more than once
Denial of Service	Attacker floods a service with requests at a rate far too fast for it to respond 	Reject traffic from an address when it becomes clear that it has initiated an attack

PolarLake

www.polarlake.com

even more compelling.

To understand this, it is best to first break down the costs associated with managing Web services integrations. First, there are the costs of setting up valid credentials: creating and distributing client certificates, distributing information about how to access the services, and testing and debugging initial connections. Then there are the ongoing costs of maintaining keys and certificates, adding new clients, managing logs, upgrading applications, and fixing problems as they arise.

Service firewalls can significantly reduce both initial and ongoing management costs. First, service firewalls make it easy to set up secure communication with partners by separating security functionality from application logic. This allows the two to be tested and configured independently. Application interoperability can be established first without security, and then the functionality of the firewall can be enabled one step at a time until the message pathway is fully secure between client and firewall.

Second, a service firewall allows easy maintenance of security and services because it does not require each application to be retested after changes are made. For example, if security were built into an application, even a simple change such as allowing a new partner to send a SOAP message without a digital signature would require it to be recompiled to allow the new partner to omit the signature, and would have to undergo a full QA cycle to be requalified. By contrast, if security were performed by a service firewall, an administrator could simply make the change and deploy the new security policy in a matter of minutes because the applications are unaffected. The costs of implementing or purchasing a service firewall can be recouped through less costly management over its lifetime, especially in situations in which access control requirements or standards are constantly changing.

Conclusion

While the trade press and industry analysts have consistently bemoaned the state of Web services security, the service firewall offers a useful architecture that enterprises can deploy today to more easily and cost effectively secure and control their inbound and outbound Web services traffic, and extend tomorrow to stay current and remain flexible. But while attention has focused on security enforcement, enterprises require equal attention to the ability to manage security policy, monitor exceptions, and provide an audit trail that can be used by compliance and business managers alike.

Author Bio

Michael Hanson is the lead architect of Reactivity's secure networking products. Michael represents Reactivity in the WS-I standards organization and is a lead contributor for security and testing issues. Previously, he designed and architected secure data-center management software for CenterRun Inc., and a distributed collaboration platform for Zaplet, Inc. Michael was a member of Apple Computer's Advanced Technology Group. He holds bachelor's and master's degrees in computer science from Stanford University.

MICHAEL.HANSON@REACTIVITY.COM

LISTING 1

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap
  -envelope" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
 <S:Header>
  <wsse:Security>
   <wsse:BinarySecurityToken ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary" wsu:Id="X509Token">
    MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
   </wsse:BinarySecurityToken>
   <ds:Signature>
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.
        w3.org/2001/10/xml -exc-c14n#"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2000
       /09/xmldsig#rsa-sha1"/>
     <ds:Reference URI="#myBody">
      <ds:Transforms>
       <ds:Transform Algorithm="http://www.w3.org/2001
         /10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>EULddytSo1...</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
     BL8jdfToEb1l/vXcMZNNjPOV...
    </ds:SignatureValue>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse: ReferenceURI=" #X509Token "/>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
  </wsse:Security>
 </S:Header>
 <S:Body wsu:Id="myBody">
  <tru:StockSymbol xmlns:tru="http://www.fabrikam123.com/
     payloads">
   QQQ
  </tru:StockSymbol>
 </S:Body>
</S:Envelope>
```

LISTING 2

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-
   envelope" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
 <S:Header>
  <wsse:Security>
   <xenc:EncryptedKey>
    <xenc:EncryptionMethod Algorithm="..."/>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier EncodingType="wsse:Base64Binary"
        ValueType="wsse:X509v3">
    MIGfMa0GCSq ...
      </wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>...</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#bodyID"/>
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
  </wsse:Security>
 </S:Header>
 <S:Body>
  <xenc:EncryptedData Id="bodyID">
   <xenc:CipherData>
    <xenc:CipherValue>...</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </S:Body>
</S:Envelope>
```

Download the Code
www.sys-con.com/xml

XML for Financial Services

www.worldrg.com/fw332

WRITTEN BY **AYESHA MALIK**

XML in the Financial Services Industry

Applications of the standards

The financial services industry spends billions of dollars on IT development to maintain its competitive edge. Banks, risk management firms, and insurance companies have been focusing on XML as a means to exchange their data and to streamline the trading, settlement, and risk management processes. In light of such goals, industry consortiums have been signaling their commitment to XML by creating XML standards for data exchange.

This article will discuss the application of XML in building a risk management system. Topics covered include FpML (Financial products Markup Language), how to extend XML standards, XML Key/KeyRef for referenctial integrity, XQuery, and Web services.

I will explore the application of FpML using a case study on building a corporate risk management system. Currently, FpML is used internally by a large number of prominent financial institutions for their risk management and derivatives systems. It is one of the key standards used in financial services along with FIX (Financial Information Exchange protocol). This case study will illustrate how FpML is used in the different steps necessary to build a successful risk management system.

The Risk Management System

Financial institutions such as Citibank and JPMorgan Chase trade in many different markets across the globe. Each trading department keeps track of how much it has invested and how sensitive the investment is to changes in market conditions. For instance, say the Hong Kong office invests $30 million in a financial security such as a currency derivative. The contract is structured in such a way that if interest rates rise by 0.1%, it will lose $10 million. Or say the Hong Kong office holds a contract with a firm whose credit rating has fallen to DDD, according to Standard & Poor's. If it does not get out of the contract soon, the office runs a credit risk of losing $7 million in case the counterparty cannot pay its part of the agreement. By closely monitoring market and counterparty conditions, each local office calculates the quantitative risk it takes by conducting sophisticated statistical tests that return risk metrics and allow it to change its trade position if required.

Similarly, the Dubai, London, and Buenos Aires offices conduct the same analysis for their investments. For each local office, this is a very good way of managing risk. However, it's potentially dangerous when you consider the situation in which all four offices will lose $10 million each if interest rates go up by 0.1%. That kind of loss can bankrupt a small firm. Hence the need for corporate risk management.

A corporate risk management system keeps the aggregate trading positions of all its domestic and overseas offices and runs analytics to monitor the risk exposure of the firm as a whole and ensure that it doesn't exceed a conservative threshold. In other words, each trading unit sends the head office its investment statistics: what it bought, how much it bought, and who it bought from. The head office then gets market and counterparty credit information from a data vendor such as Reuters and estimates its risk exposure. Such a large-scale software system is referred to as a corporate risk management system and is incredibly important for companies who trade in the trillion-dollar market of financial derivatives.

Figure 1 illustrates the workflow of a corporate risk management system.

I have discussed the business workflow in detail. What about the architecture of such a system? The main thrust

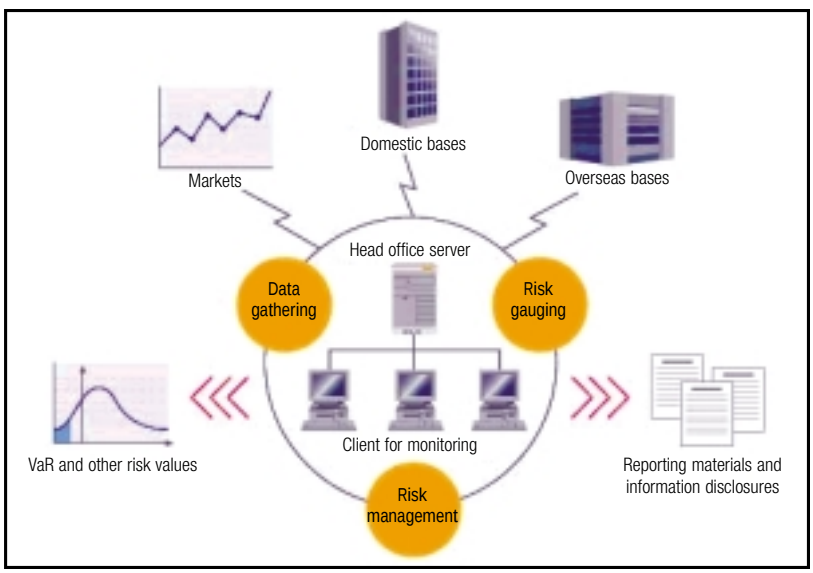


Figure 1 • Risk management workflow (Source: CUBE SYSTEM, Inc.)

AUTHOR BIO

Ayesha Malik is a senior consultant at Object Machines, a software engineering firm providing Java technology and XML solutions to businesses. She serves on the Architecture Working Group of Financial Products Markup Language (FpML), a data-interchange standard set forth by International Swaps and Derivatives Association (ISDA). Ayesha holds a BA with honors from Harvard University and an MS from Columbia University, where she studied operations research, applied mathematics, and computer science.

of the system involves the messaging of data to the head office and the querying of this data for analytics purposes. In addition, market and counterparty data is messaged from a third-party data vendor such as Reuters. The discussion of this article consists of an analysis of the format of this data. For derivatives, the XML standard of choice is fast becoming FpML.

Modeling Derivatives Using FpML

FpML's (www.fpml.org) working draft for Version 3.0 is available online and contains an XML Schema version of the standard. The FpML schema architecture consists of three main categories: trade, portfolio, and party (see Figure 2). The trade complex element is the main part of FpML and contains all the terms and conditions of a trade. Currently, FpML covers the following asset classes: interest rate derivatives, forward rate agreements, FX, and equity derivatives. Figure 3 shows a partial FpML instance document for an equity option. The trade complex type contains information such as trade date and all the particulars of the contract for an equity option such as strike price. The party complex type contains information such as party name.

The FpML organization consists of working groups for the study and construction of appropriate schema models for derivative instruments. The members of these working groups are technical architects, industry experts, and representatives of the world's leading financial institutions. They have constructed flexible schemas that can incorporate a wide variety of derivatives models.

Each trading department must populate the FpML instance document relevant to the instrument that they have traded. Usually, traders enter all the details of their trades in a user interface that is connected to a database. In order to send this information to the head office, developers must construct the FpML instance document by populating it from the database, ensure that it is a valid instance document by validating it against the FpML schema, and then send it to the head office's risk management system.

If the system is using Java, then the FpML instance document can be constructed using JDOM. Given that the FpML schema tries to adhere to an object-oriented framework and use types for concepts that represent the same meaning, it makes sense to use JDOM classes. However, I must caution that FpML and other financial standards are still a work in progress. As more asset classes are added, it becomes important to control types and OO design, and the standards committees must have a quality control committee to maintain architectural integrity through all parts.

Extending the FpML Model

In discussions of the practical applications of an industry standard, it's common to come across a situation when the industry standard does not address the company's needs. For instance, corporate risk management requires counterparty information to determine whether the counterparty to the contract can meet its end of the bargain. Such an analysis is known as credit risk management and is an important component of gauging risk exposure. In FpML, the structure for counterparty information is very weak – it only includes partyId and partyName – and it's necessary to extend the standard in order to meet the full needs of our system (see Figure 4).

For credit risk analysis, the system must consider the risk ratings, such as Moody's rating, for the counterparty and the counterparty's parent. In addition, it is a good idea to know whether a corporate action such as a merger and acquisition has affected the counterparty in any way.

The best way to extend the standard is to define your extensions in your company's namespace so that it's always easy to separate the standard from additions made internally. This approach involves qualifying the elements by their namespace codes. As a hypothetical example, say BigBank's offices are using FpML to send information to the risk management system. They want to send counterparty information as well as the financial instrument information. BigBank extends the FpML party component as shown in Listings 1–3 and gives the new schemas to its offices to use to create their instance documents. Note that the namespace prefix for BigBank's additions is BB and this is used both to reference the risk rating complex type and also to qualify BigBank's definitions in the instance document.

The methodology described can be used whenever extending industry standards. If the industry covers your needs one day, you can simply replace your additions with the industry standard additions.

Improving Data Quality

In the previous section, I discussed the importance of having details on counterparties for credit risk analysis. When a trader enters into a trade agreement, he or she has only the counterparty name and this is the information that comes to the head office. Optional elements such as Moody's risk rating are left empty and must be filled by the head office itself. Risk rating information, along with information regarding parentage hierarchies and corporate actions, can be found from sources such as Bloomberg, Reuters, and other news agencies.

Traditionally, companies have designed

What Are Derivatives?

Derivatives consist of options, futures, forwards, and swaps, instruments that derive their value from the values of other underlying assets, such as stocks, bonds, currencies, and commodities (hence the name derivative). Derivatives are also known as over-the-counter instruments since they are privately negotiated instead of traded at a public exchange. Because of the complexity and leveraging power of these instruments, risk management and monitoring becomes essential.

Derivative trading and investment has recently become a booming dynamic market whose participants include all major companies and investors in the world. At the end of 1999, the total estimated notional amount of OTC derivatives outstanding worldwide was approximately $88.2 trillion, with 10% growth in 1999 alone.

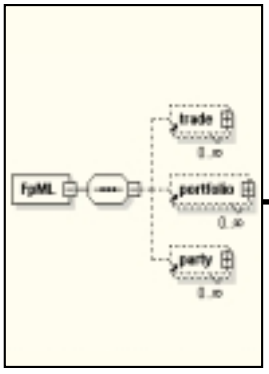


Figure 2 • FpML top level (Source: FpML)

```
<?xml version = "1.0" encoding = "UTF-8"?>
<FpML>
  <trade>
    *
      <tradeDate>2001-10-23</tradeDate>
    *
    <equityOption>
      <productType>americanCallStock</productType>
    *
      <strike>
     <strikePrice>8.00</strikePrice>
      </strike>
    *
    </equityOption>
  </trade>
  <party>
    *
    <partyId>CITIUS33</partyId>
    *
  </party>
</FpML>
```

Figure 3 • Partial FpML instance document

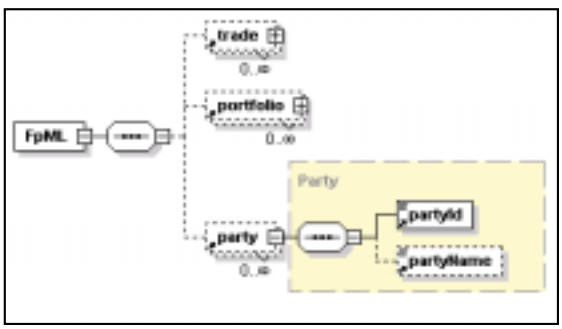


Figure 4 • FpML party coverage (Source: FpML)

their own software components that connect to the Reuters data stream, for instance, and store and parse the information for their own needs. These days, a crop of companies that come under the generic label of third-party data aggregators is becoming prevalent. Our hypothetical BigBank would send its FpML instance document with the extended party component to the data vendor and ask it to fill in any missing information. The data vendor returns the cleansed and augmented data to the head office, which then uses the party details to run its credit risk analysis.

The importance of market and counterparty information in calculating risk cannot be emphasized enough. Data quality is essential when constructing risk metrics since bad data can lead to huge miscalculations and potential loss-

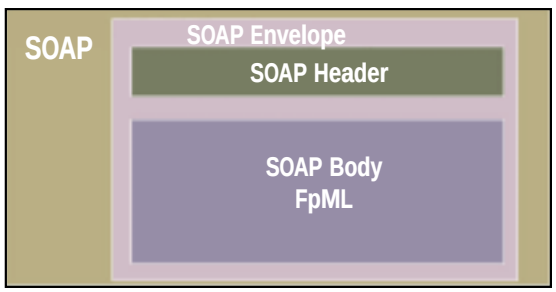


Figure 5 • Parts of SOAP

es for the company. In order to effectively exchange information, the data vendor and BigBank must agree on a data format and especially on the meaning of each of the tag names. Industry standards (and their extensions) are usually the most cost-effective choice for a company using a data vendor because it can leverage the fact that the vendor uses the same standard for several of its clients.

Using Key and KeyRef for Referential Integrity

Keeping track of references is very important when you're dealing with instruments and their counterparties. Later, you might want to link up portfolios (a book of trades) to a counterparty or to a trading unit. In such cases, the ability to maintain referential integrity is essential for proper bookkeeping and accurate risk management. In XML Schemas, use KeyRefs (IDREFs are a similar concept used for DTDs) to achieve this functionality. Different XML financial standards will adhere to different methods of interlinking documents and enforcing uniqueness and association. FpML, for instance, is considering using IDREFs or Key/KeyRef for its next release. If you are using schemas and are extending any part of the standard yourself, you should try to use keys and KeyRefs for this purpose.

Consider, for example, trying to link a financial instrument with a particular counterparty. The ability to have this foreign key relationship is illustrated by using KeyRef in the following manner. The KeyRef element specifies that an attribute or element value correspond to those of the specified key or unique element. In our example, consider that each trade committed is done with a counterparty. To link a trade to a counterparty, both must contain the unique party identifier. Listings 4 and 5 show how this is achieved. Since XPath expressions are used, the parser will only validate if the instance document has the same key and KeyRef.

A key is defined to specify unique instances of a partyId and a KeyRef is used to link up the partyId under the Trade complex element to the partyId under the Party complex element. Keys and KeyRefs are similar to primary and foreign keys in traditional database management systems. This ensures that you have the same counterparty in the trade and in the party associated with that trade.

Using XQuery to Query FpML Instance Documents

Once the FpML instance documents have been constructed, the data is ready to be handed over to risk analytics systems. Companies may buy sophisticated risk analytic systems or build them in-house. In either case, the risk engine will not necessarily be able to process an FpML instance document (even though it is often the case that once an industry standard becomes prevalent, increasing numbers of vendors incorporate the ability to interface with the standard). In such cases, you might have to query your instance document to get the information that the risk analytic engine requires. XQuery is still in the Recommendation phase but is a powerful method of querying XML documents.

XQuery allows you to create a new XML result document by using certain expressions. The example that follows shows how you can get the risk rating of a counterparty for a particular credit risk metric calculation. I find using XQuery a cleaner and easier method of querying instance documents than using XSLT transformations. XQuery is instinctive for developers familiar with SQL since it's based on the same concepts.

For the party in the instance document in the last section, I could request the party name and the Moody's rating under the heading party because this is the requirement of my risk analytics engine and this is how it expects the information. This is a simple example to illustrate the beauty and simplicity of XQuery. Note that you need an XQuery engine for this to work. One open-source Java-based engine is available at Fat Dog (www.fatdog.com).

```
<party>
{
    for $p in document("http://www.
     fpml.org/party.xml")/FpML/party
    return
            { $p/riskRating }
}
</party>


<party>
        <riskRating>
         <Moodys>AAA</Moodys>

<StandardAndPoors>BBB</StandardAndPoors>
        </riskRating>
</party>
```

Using Web Services and SOAP for Messaging FpML

Now that we've explored all the workflow issues of building a risk management system, it is good to consider the messaging architecture this information will use for transportation. In this article, I advocate the use of Web services and SOAP as the messaging infrastructure for this transport. According to Gartner, Web services–related architectures result in a 30% increase in the efficiency of IT development projects.

Web services is architecture for connecting modular software components that can be invoked by XML messages over, for example, HTTP or another message transport standard such as TIBCO. SOAP is one of the primary protocols of the Web services architecture. SOAP is a protocol specification and defines a uniform way of passing XML-encoded data and for communication between software applications. SOAP itself is an XML document that consists of the following parts: Envelope element, Header element, and Body element. In the case where we are using SOAP to carry our FpML document, we would put FpML in the Body element of our message (see Figure 5).

In some cases, the XML standard will include message information in its content, as is done in the case of ebXML. FpML plans to introduce a messaging framework to guide users in capturing

the FpML instance document with messaging information.

Summary

In this article, I have explored the business and system workflow of a corporate risk management system. I have shown how different business requirements are met using one particular XML standard for financial services and various XML technologies. This includes how to model derivatives using FpML and how to extend the standard in case your business needs exceed the XML model provided by FpML. I have discussed how XML standards are necessary to leverage economies of scale savings when getting improved data quality from a data vendor. The W3C is also constantly introducing and refining technologies for making XML a true data repository. Two examples of this functionality are XQuery and Key/KeyRefs, which mimic the advantages of SQL and relational databases. Finally, there is a brief discussion of the Web services architecture that is the transport protocol of the data.

XML standards for the financial services industry are becoming important for interoperability and efficiency in the industry. According to a *Wall Street and Technology* survey published in May 2002, 81% of leading financial services organizations see XML as absolutely critical to their business success. FpML is one of the leading standards for modeling derivatives products and the one practical application of it lies in a risk management system. If you're building or reengineering your risk management system, both XML technologies and XML standards are available and suitable for efficiency and interoperability.

References

- van der Vlist, Eric (2000). "XML Linking Technologies." XML.com. October.
- *Financial products Markup Language (FpML):* www.fpml.org
- *XML Query Requirements:* *www.w3.org/TR/xmlquery-req*
- *XML Schema Part 1: Structures*: www.w3.org/TR/xmlschema-1
- *XML Query Use Cases:* www.w3.org/TR/xmlquery-use-cases

AYESHA.MALIK@OBJECTMACHINES.COM

LISTING 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.bigBank.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.bigBank.com" elementFormDefault=
  "qualified">
 <xs:element name="riskRating">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Moodys" type="xs:string"/>
    <xs:element name="StandardAndPoors" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

LISTING 2

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.fpml.org/2002/
  FpML-3-0"
xmlns:BB="http://www.bigBank.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.fpml.org/2002/FpML-3-0" elementForm
  Default="qualified" attributeFormDefault="unqualified">
 <xsd:import namespace="http://www.bigBank.com"
   schemaLocation="bigBank.xsd"/>
<xsd:element name="FpML">
  *
 *
<xsd:complexType name="Party">
  <xsd:sequence>
   *
   <xsd:element ref="partyName"/>
   <xsd:element ref="BB:riskRating"/>
  </xsd:sequence>
 </xsd:complexType>
         </xs:sequence>
      </xs:complexType>
*
*
   </xs:element>
</xs:schema>
```

LISTING 3

```
<?xml version="1.0" encoding="UTF-8"?>
<FpML xmlns="http://www.fpml.org/2002/FpML-3-0"
xmlns:BB="http://www.bigBank.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fpml.org/2002/FpML-3-0>
 *
 *
<party>
 *
  <partyName>Citicorp</partyName>
  <BB:riskRating>
   <BB:Moodys>AAA</ BB:Moodys>
   <BB:StandardAndPoors>BBB</ BB:StandardAndPoors >
  </BB:riskRating>
 </party>

 *
 *
</FpML>
```

LISTING 4

```
<?xml version="1.0" encoding="UTF-8"?>
<FpML xmlns="http://www.fpml.org/2002/FpML-3-0"
xmlns:BB="http://www.bigBank.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fpml.org/2002/FpML-3-0>
 *
 *

 <trade>
  <xsd:element name="partyId">
   <xsd:complexType>
    <xsd:attribute name="idref" type="xsd:integer"
     use="required"/>
   </xsd:complexType>
  </xsd:element>
 *
 *
 </trade>

 *
 *
 <party>
  <xsd:element name="partyId">
   <xsd:complexType>
    <xsd:attribute name="id" type="xsd:integer"
     use="required"/>
   </xsd:complexType>
  </xsd:element>
 *
 *
 </party>

 *
 *

 <xsd:key name="partyId">
  <xsd:selector xpath="./party/partyId"/>
  <xsd:field xpath="@id"/>
 </xsd:key>
 <xsd:keyref name="partyIdref" refer="partyId">
  <xsd:selector xpath="./trade/partyId"/>
  <xsd:field xpath="@idref"/>
 </xsd:keyref>
</FpML>
```

LISTING 5

```
<?xml version = "1.0" encoding = "UTF-8"?>
<FpML>
  <trade>
    *
        <partyId idRef=01988/>
    *
</trade>
<party>
     *
     <partyId id=01988/>
     *
  </party>
</FpML>
```

Download the Code
www.sys-con.com/xml

Where OPEN Minds Meet

Linux is thriving! Financial services, telecommunications, government and retail are utilizing this amazing technology. Shouldn't you?

Attend Keynotes and learn from inspiring visionaries who are devoted to Linux and Open Source.

Wednesday, January 22 / 10:30 am - 11:30 am
Hector Ruiz, Ph. D.
President & CEO
AMD

Wednesday, January 22 / 1:30 pm - 2:30 pm

Jeffrey M. Birnbaum
Managing Director & Global Head of Enterprise Computing in the Institutional Securities Division
Morgan Stanley

Michael Tiemann
Chief Technical Officer
Red Hat

Thursday, January 23 / 10:30 am - 11:30 am
Steven A. Mills
Senior Vice President & Group Executive
IBM Corporation

Thursday, January 23 / 1:30 pm - 2:30 pm
Randy Mott
Senior Vice President & Chief Information Officer
Dell Computer Corporation

Conference: January 21-24, 2003
Expo: January 22-24, 2003

The Javits Center
New York, NY

LinuxWorld Conference & Expo

- LFS LINUX FINANCIAL SUMMIT

  All-new - Linux Financial Summit is an executive program created to address the growing demand for quality information and insight on Linux and Open Source solutions for the financial industry.

  Sponsored by:

  egenera, hp invent

- ENTERPRISE SOLUTIONS CENTER

  Check out the all-new Enterprise Solutions Center, where you'll experience multi-vendor Linux and Open Source enterprise solution demonstrations.

  Presented by:

  LINUX INTERNATIONAL, WILD OPEN SOURCE

- Participate in LinuxWorld's world-class education program and learn how Linux has been implemented in financial services, communications and government, gain technical and enterprise know-how in areas such as sys admin, business and emerging technologies.
- Explore the newest technologies from leading Linux and Open Source suppliers and a growing array of start-ups.
- Take your seat at The Golden Penguin Bowl, the lively quiz show.

Cornerstone Sponsor

hp invent

Platinum Sponsors

AMD, IBM, intel, redhat, Sun microsystems — We make the net work.

Silver Sponsors

macrovision — Globetrotter Software Division

UNITEDLINUX

www.linuxworldexpo.com

IDG WORLD EXPO

Native XML Databases Today

A look at what's available

WRITTEN BY
CHRIS HALASCHEK AND JOHN A. MILLER

The use of XML is on a rapid incline, and with this growth comes high demand for storing and retrieving XML documents in their natural, structured format. This need has resulted in many companies developing what are now referred to as native XML databases. Currently, there's a variety of these databases on the market, and still many people are unaware of their availability and function. A brief tour of a select few will familiarize you with some products that are available and give you a general idea of the functionality of native XML databases.

What Are Native XML Databases?

As the name implies, native XML databases (NXDBs) are specially designed systems used to store XML documents. What differentiates NXDBs from other databases is the way they actually store these documents. XML-enabled databases break down the XML documents into data elements for storage, possibly in relational tables. In contrast, NXDBs use XML documents as their fundamental unit of storage. This storage process parallels the way relational databases have rows in tables as their fundamental unit of storage. NXDBs conserve the original structure of the document as opposed to breaking it down into data objects or tuples. This new technique actually provides quite a few benefits.

One of the benefits is that users don't have to worry about mapping the XML document to an alternate data structure; they can simply insert the data in XML format and retrieve it in its original XML format, known as round-tripping. Round-tripping is especially valuable when very complex XML structures exist that would be almost impossible to map into a more structured database. Another benefit is that due to the storage of the document as a whole, queries don't require rebuilding of the document. Thus, users experience faster retrieval times. Finally, many NXDBs may exploit XML-specific capabilities not included in most relational databases currently available.

General Functionality of NXDBs

While all NXDBs are slightly different, the vast majority share the same basic functionalities and standards. Of course, they all maintain data at the document level. Many NXDBs manage stored documents as collections. Similar to the relational concept of a table, storing documents in this manner allows users to query and interact with the documents as a set. Many NXDBs also follow at least a subset of the XML:DB Initiative standards released by the World Wide Web Consortium (W3C).

Nearly all NXDBs support some kind of query language – XPath is the most popular query language currently used. It was recommended by the W3C and is supported in just about every NXDB. Unfortunately, XPath does have some limitations, including its failure to support joins across documents, sorting, and grouping. A more recent query language recommended by the W3C is XQuery (XPath is actually a subset of XQuery). Many of XPath's problems have been resolved with XQuery. While XPath is more widely used, due to the advantages of XQuery, most NXDBs will most likely migrate toward this language.

Almost all NXDBs also support some level of indexing. Most allow indexing on the data stored as collections, more specifically, elements and attribute values. This can greatly increase the speed of queries.

NXDBs usually support some type of updates. Currently, a variety of techniques is used. A few products support XUpdate (from the W3C XML:DB Initiative). However, most NXDBs force the entire document to be retrieved in order for changes to be made, while the XUpdate standard allows users to modify document fragments.

Many NXDBs also provide an API in which various methods are supplied for connecting and executing operations on the database. This allows a programmatic interface to the NXDB.

Various NXDBs are supplying very friendly graphical user interfaces in which clients can easily perform numerous tasks, such as view, insert, query, and update XML documents. In addition, some of the GUIs provide numerous administrative functions.

Just about all NXDBs support transactions, locking, and concurrency control. This type of support is practically a must with multiple-user databases.

As mentioned earlier, all NXDBs support some level of round-tripping. Again, this means that the user will get back the same XML document (or portion thereof) that is stored in the database.

The Tour

The products that will be surveyed here are native XML databases that support either XPath or XQuery. They also have accessible demos or evaluations. Issues such as supported standards, portability, user interface, and features will be addressed. Please keep in mind that this is not a comprehensive list, but rather a general look at a few of the products available.

Xindice

Apache Xindice is an open-source NXDB designed to store collections of small to medium-size documents. Since Xindice is written in the Java programming language, it can be run on any platform that supports the Java Development Kit. Xindice supports a variety of W3C standards, including the XPath query language and the XUpdate language for database updates. Xindice also provides developers a standard Java API that supports the W3C standard XML:DB API.

For administration purposes and direct interaction with the database, Xindice provides a text (command-line tool) interface. In addition, Xindice allows indexing at the element and attribute levels of XML documents. Xindice has a few added features, including a CORBA API, as well as an XML-RPC plug-in that supports access from various languages such as Perl, AppleScript, and PHP. Finally, Xindice has included XMLObjects, which allows users to add extra functionality to the server.

TEXTML Server

TEXTML Server is a commercial NXDB created by IXIASOFT, which claims to have produced the first native XML database for OEMs and developers of XML applications. TEXTML Server is designed to run on the Windows platform. The server side of TEXTML is compatible with Windows NT 4, 2000, and XP. Additionally, the client side is compatible with Windows 98 and ME. TEXTML Server supports the W3C standard XPath for its query language. To administer the database, TEXTML Server provides a graphical user interface. This tool allows you to perform various operations, such as setting up indexes, building document bases, and managing security.

TEXTML Server has also added a variety of features. Indexes can be set up on both element and attribute values. There are COM+ and Java APIs to aid in the development of client applications. TEXTML Server also allows very high-performance searches in which users can search on string, date, time, full-text, and numerical indexes. TEXTML Server allows sorting of results and provides transaction management. It also includes an ASP toolkit for the rapid development and deployment of client applications. Finally, TEXTML Server provides XML Spy, XMetal, and WorX integration toolkits so that the database can be used as the primary repository for documents edited with these popular applications.

Db4XML

Db4XML is an open-source NXDB developed at the University of Georgia. Db4XML supports both XPath and XQuery for its query language. Db4XML is a main memory database, which allows for increased performance. Because Db4XML is written in Java, it can be deployed on any operating system that supports the Java Development Kit. Db4XML has a graphical user interface that supports various administrative and interactive functions. Database users can run queries, add documents, add schemas in either DTD or the XML Schema format, and perform various other administrative functions. Db4XML also utilizes indexing to speed up database performance. Examples of index types available are both path and value indexes. In addition, Db4XML provides a Java API for application development. Db4XML supports transaction management and full recovery in the event of a system crash.

"Due to the way in which NXDBs are built from the ground up, some of the relational giants would have to go to great lengths to match the performance of NXDBs"

Cerisent XQE

Cerisent XQE is a commercial NXDB developed by Cerisent. Cerisent XQE is capable of managing large sets of XML elements, on the order of 10GB per single-CPU server. Cerisent XQE supports the XQuery and XPath query languages in addition to the W3C XML Schema definition standard. Cerisent XQE is supported by a variety of platforms; it can be run on Windows 2000 and XP, Linux, and the Sun Solaris platform. Cerisent XQE has numerous features added in as well. The database is fully transactional and allows for simultaneous loads, queries, and backups – it also allows for fragment-level updates. Cerisent XQE has the capability to index on all element names, attribute names, element content, and attribute values.

eXtensible Information Server (XIS)

XIS is a commercial NXDB developed by eXcelon. XIS includes a native XML database, an XSLT engine, and a comprehensive suite of tools that perform various functions. XIS supports multiple standards, including XML 1.0, XSLT, XML Schema, DOM level 1 and level 2 APIs, and the XQuery and XPath languages. XIS is also supported by various platforms, including Windows NT 4, 2000, XP; Solaris 2.6, 7, and 8; HP-UX 11; and Red Hat Linux 7.1. The core of XIS is its Dynamic XML Engine (DXE). This engine includes a DOM and file system interface, document linking, binders, a Lightweight Directory Access Protocol (LDAP) manager, a configuration manager, indexing, triggers, updates, an XSLT processor, and a Java API. XIS is also capable of integrating data from legacy sources through its XConnects Integration Engine. XConnects thus provides a solution for combining data and content sources. Another feature of XIS is its DXE Manager, which is a graphical administration tool that includes a variety of functions. The DXE Manager contains an interface for browsing and managing DXE repositories, an update and query wizard, cache and route configuration options, the ability to manage online backup and restoration of DXE repositories, a data migration tool, and a bulk loading tool used to load bulk directories of XML data. In addition, XIS includes a Map Designer, which is a visual tool used to define mappings from back-end sources into XML or vice versa. XIS also includes transactional support and has a Java API for application development. Finally, XIS includes another graphical tool, Stylus Studio, which is a development environment for XML-based applications. Stylus Studio allows for project management and includes a Java source editor, a DTD and XML Schema editor, an XSLT editor, an XML-to-XML mapper (used to define mappings between different XML schemas), and a debugger.

"NXDBs conserve the original structure of the document as opposed to breaking it down into data objects or tuples"

Infonyte DB

Infonyte DB is a commercial NXDB developed by Infonyte. The database is based on Infonyte's PDOM, a persistent implementation of the W3C DOM API for XML. Infonyte DB supports a wide variety of standards, including the W3C DOM level 2, XPath, XSLT 1.0, XQL '99 with some extensions, and the Java API standard for XML processing. Since Infonyte DB is completely written in Java, it is platform independent.

Infonyte DB also has a graphical user interface, the XML Workbench, which supports functions such as creation, modification, exploration, querying, and transformation of documents in the Gigabyte range. This tool also includes a text pane for viewing and editing documents, a tree view for navigation of documents, an HTML viewer, a query facilitator, and user interfaces for maintaining database settings. Along with the XML Workbench, Infonyte has developed PXSLT, which is an XSLT processor. Infonyte DB allows for text and data indexes, document collections, and full update capabilities.

X-Hive/DB

X-Hive/DB is a commercial NXDB developed by X-Hive Corporation. The database is designed to scale in terms of concurrent users and database size and is also capable of storing media objects, such as images, sounds, and Office documents. X-Hive Corporation has included support for many W3C standards, including XML 1.0, XQuery, XPath, XPointer, XLink, XSL, DOM, WebDAV, and J2EE. X-Hive is supported across many platforms, such as Red Hat Linux 6.2 and 7, Sun Solaris 7 and 8, and Windows 2000. X-Hive/DB also has a graphical administrative interface for database management. This interface allows for database manipulation and configuration. The database has a built-in transaction mechanism and supports load balancing and replication of databases over multiple machines. Also supported are various indexing methods. These methods include library, element name, ID attribute, value, context conditioned, and full-text indexes. X-Hive/DB also has a built-in XSL engine for transformation of XML documents to other formats and publishing of XML data in HTML or PDF. The database has built-in interfaces included as well. The X-Hive/DB Java API contains methods for storing, querying, retrieving, transforming, and publishing XML data. In addition, there are an interface to relational databases, bridges to XML editors and full-text search engines, and support for J2EE and WebDAV. X-Hive/DB has a versioning mechanism for tracking differences within XML data.

Ipedo XML Database

Ipedo XML Database is a commercial NXDB developed by Ipedo. Ipedo XML Database supports the XML Schema, XQuery, XPath, and XSLT W3C standards. Ipedo's implementation of XPath has been extended to allow queries across multiple documents in a collection. Ipedo XML Database is written in Java, thus it is supported across multiple platforms, including Sun Solaris 7 and 8; Red Hat Linux 6 and 7; and Windows NT, 2000, and XP. This database also offers a graphical interface that's used to perform various database-management functions. In addition, Ipedo XML Database allows for a variety of index types on the collection level, allowing the database administrator to tune the performance of the database. Ipedo also permits concurrent user access and lock management. The database supports a variety of APIs as well. These include Java and EJB APIs for J2EE applications, as well as COM and SOAP APIs for Microsoft .NET applications. The database also includes support for WebDAV-enabled clients and applications. Finally, Ipedo XML Database has graphical Schema and Query managers that are used for database interaction.

NeoCore XMS

NeoCore XMS is a commercial native XML database management system developed by NeoCore. The data-

CTIA Wireless 2003

www.ctiashow.com

Database Name	Windows OS	Alternate OS	XPath*	XQuery*	XUpdate*	Graphical Tools	Command Line Tools	APIs	XSLT Engine
Xindice	✔	✔	✔		✔		✔	✔	NA
TEXTML Server	✔		✔			✔		✔	NA
Db4XML	✔	✔	✔	✔		✔		✔	NA
Cerisent XQE	✔	✔	✔	✔		NA	NA	NA	NA
eXtensible Information Server	✔	✔	✔	✔		✔		✔	✔
Infonyte DB	✔	✔	✔			✔		✔	✔
X-Hive/DB	✔	✔	✔	✔		✔		✔	✔
Ipedo XML Database	✔	✔	✔	✔		✔		✔	NA
NeoCore XMS	✔	✔	✔	✔		✔		✔	NA
Tamino XML Server	✔	✔	✔	✔		✔		✔	NA
GoXML DB	✔	✔	✔	✔		✔		✔	NA

**: W3C Standard*

NA: Information not available

base is built on NeoCore's patented Digital Pattern Processing technology. NeoCore XMS supports both the XPath and XQuery languages. NeoCore XMS is also compatible with multiple platforms, including Solaris 2.8 64-bit and Windows 2000 Advanced Server, Server, and Professional. The database has a Web-based management console for administrative and interactive purposes as well.

NeoCore XMS supports multithreading, access control, and transactional support. The database system also has various interfaces for application interaction, including a Java class interface, a C++ API, a Microsoft COM API, an HTTP API, and an EJB interface for J2EE application servers. In addition, NeoCore XMS has a complete data migration toolkit. NeoCore XMS also supports queries across multiple documents; returns well-formed XML; and will return documents, document fragments, and elements.

NeoCore XMS has a variety of tools and plug-ins. The database currently includes NeoCore Insight, a graphical discovery tool that allows users to analyze data and its inherent relationships, create dynamic and variable views, reveal patterns and correlations within the data, and create graphs and gauges to illustrate the importance of data. NeoCore XMS also provides support for Data Junction, which allows Data Junction's Export, Translate, and Load (ETL) engine to import and export data in various formats. Finally, NeoCore XMS has released the NeoCore XML Spy Integration Kit. This kit provides the option of using Altova's XML Spy to manipulate data stored in the database.

Tamino XML Server

Tamino XML Server is a commercial NXDB developed by Software AG. This database supports multiple W3C standards, including XML, XML Schema, XSL, XSLT, DOM, JDOM, SAX, XPath, XQuery, and SOAP. The database is available on all major operating systems. Tamino XML Server has multiple features and tools that assist in the management and use of the database. Tamino X-Tension is a service that allows users to write extensions that customize server functionality. Another feature, Tamino Manager, is an administration tool that provides a graphical user interface that runs on standard Web browsers. It's essentially used to manage the entire system through the Web. The Security Manager is a Web-based tool used to control both the access rights and the security of the database. X-Port is a Web server interface that provides data transfer through various standard HTTP servers. Tamino X-Plorer is very similar to the Tamino Manager but is used as a navigation tool for documents stored in the database. In addition, X-Application provides JSP tags to embed access to the database through Web pages.

Tamino XML Server accepts well-formed XML documents as well as any other type of non-XML data, such as audio, video, and PDF documents. Finally, the system supports multiple APIs, including an EJB API for integration with major application servers, and a Java API, which supports DOM level 2, JDOM, and SAX.

GoXML DB

GoXML DB is a commercial database developed by XML Global. GoXML DB supports both XPath and XQuery for its query language. GoXML DB is also supported across multiple platforms, including Windows 98, NT, 2000, ME, and X; Solaris; and Linux. The database has a graphical interface used for database interaction and management. Some of this tool's functions include querying data, managing schemas and documents, and creating indexes. The database also supports the update extensions of XQuery. Thus, users can insert, update, and delete at the document, attribute, and element levels. GoXML DB provides a graphical schema manager used to store and organize various schema, as well as full text indexing and searching. Finally, for Web-based data entry, XML Spy's e-Forms has been integrated with GoXML DB.

Conclusion

As you can see, a multitude of NXDBs are available on

the market. Keep in mind that they are by no means limited to the select few discussed here. The large number of NXDBs triggers a few thought-provoking questions: Which is the best native XML database for me? What is the future of the market?

Which NXDB is best for you? That depends on your specific needs and direction. Clearly, almost all products support the same basic standards and perform similar functions. Due to their similarities, many developers have added special features and plug-ins to distinguish their product from the crowd. Therefore, you need to determine what it is you essentially wish to gain from your database and decide which product might best meet your requirements. Just about every NXDB product on the market has an evaluation license and/or demo for thoroughly exploring the features offered, which enables you to make an informed decision on which product to use. Conduct more in-depth surveys of various products and then make an informed decision.

You might also wonder whether NXDBs will just be a niche in the database community or will really cut into the realm of large, popular relational databases. Due to the way in which NXDBs are built from the ground up, some of the relational giants would have to go to great lengths to match the performance of NXDBs. As we all know, anything is possible…but it's apparent that there's a need for XML databases. In the constantly expanding world of XML and its applications, data will have to be stored. So regarding the future of the market, the potential for growth is clearly there – only time will tell.

References

- Bourret, Ronald. "XML and Databases." www.rpbourret.com/xml/XMLAndDatabases.htm
- Bourret, Ronald. "XML Database Products." www.rpbourret.com/xml/XMLDatabaseProds.htm
- *Xindice:* http://xml.apache.org/xindice/index.html
- *TEXTML Server:* www.ixiasoft.com
- *Db4XML:* http://maxwell.cs.uga.edu/~xmldb
- *Cerisent XQE:* www.cerisent.com/cerisent-xqe.html
- *eXtensible Information Server:* www.exln.com/products/xis
- *Infonyte DB:* www.infonyte.com/en/index.html
- *X-Hive/DB:* www.x-hive.com
- *Ipedo XML Database:* www.ipedo.com/html/products_xml_dat.html
- *NeoCore XMS:* www.neocore.com/products/products.htm
- *Tamino XML Server:* www.softwareag.com/tamino
- *GoXML DB:* www.xmlglobal.com/prod/db

Author Bios:

Chris Halaschek graduated magna cum laude with a BS degree in computer science from the University of Georgia. Chris develops handheld applications for Edelweiss Systems, LLC, where he has been the team lead for numerous projects involving Palm OS and Pocket PC platforms.

John A. Miller is a professor of computer science at the University of Georgia. John received a Ph.D. in information and computer science. He is an associate editor for ACM Transactions on Modeling and Computer Simulation *and IEEE* Transactions on Systems, Man, and Cybernetics.

chalaschek@hotmail.com

jam@cs.uga.edu

WRITTEN BY **HITESH SETH**

Java Architecture for XML Binding

Efficient mapping between Java and XML documents

Two foundation technologies, Java and XML, represent the marriage of portable code and data. A key ingredient of a successful marriage is compatibility. XML and a number of XML-based vocabularies are being used extensively as the standard data-exchange mechanism (and beyond) by both stand-alone Java 2, Standard Edition (J2SE) applications and server-based Java 2, Enterprise Edition (J2EE) enterprise applications.

A basic requirement in a number of these programmatic scenarios is the mapping of XML data as Java objects and/or vice versa (converting Java objects into XML). Although there exists a core set of APIs for processing XML, particularly the Java API for XML Processing, which includes basic DOM/SAX-based XML parsing capabilities, object-oriented Java programmers have always needed to consume, manipulate, and generate XML in a more object-oriented fashion through a standardized API set and utilities. The requirement is similar to those presented in scenarios where relational data must be mixed with Java objects, which has resulted in a number of object-relational mapping mechanisms.

Also highlighting the need for an efficient XML binding mechanism are the two extremely different mechanisms that SAX and DOM present for efficiently processing XML. Whereas SAX (Simple API for XML) represents a very efficient, albeit low-level, event-based XML processing engine, DOM (Document Object Model) provides a much easier generic in-memory, tree-based navigation structure. In a nutshell, choosing DOM over SAX may result in significant memory impact, and choosing SAX may present significant programmatic complexity.

JAXB

Java Architecture for XML Binding (JAXB, also known as Project Adelard by Java developers who have been following this project for some time) provides an efficient bidirectional mapping between Java objects and XML documents. Compared to SAX, JAXB provides ease of programming, utilizing a standardized Java objects–based binding framework and a set of tools that can convert existing XML Schemas to JAXB binding classes. However, JAXB-generated Java classes aren't generic parsers that consume any type of XML, but a specific set of XML documents as represented by the underlying XML Schema. This results in a highly efficient mechanism, since the resulting content tree doesn't have the overhead of generic tree-manipulation logic. Figure 1 shows the basic architecture of the JAXB programming model.

JAXB is particularly useful when the processing process is aware of the various XML Schema structures (for known XML documents). Typical uses include manipulating XML-based configuration files, traversing large XML feeds without DOM-based memory overhead, validating user-input data, and so on.

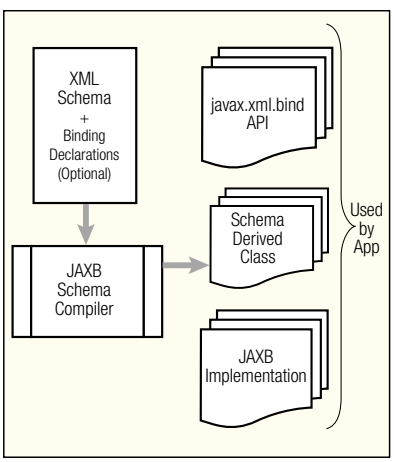


Figure 1 • JAXB programming model

Typical usage of the JAXB framework requires that the developer take an existing XML Schema–based data definition and then, using a schema compiler utility, convert it to a set of Java classes that represent the schema. JAXB, now in its Public Draft 2 status, provides an automated document-type mapping for a subset of XML Schema–based definitions. The implementation can be used to map both simple and complex XML Schemas. Table 1 shows the primitive data-type mapping between XML data and the resulting Java data types.

Key Highlights

Technically speaking, JAXB represents a set of technologies that provide the capability to:

- Generate Java classes from XML Schemas
- Unmarshal XML documents into native Java objects
- Marshal Java objects into XML documents
- Provide optional on-demand validation of XML documents

Key highlights of the JAXB framework are that it:

- Is easy to use
- Is customizable through the addition of xsd:annotation elements in XML Schema
- Hides the complexity of DOM/SAX APIs
- Uses W3C XML Schemas as the definition format (through a defined subset)
- Doesn't support DTDs directly
- Is portable (A JAXB application

AUTHOR BIO

Hitesh Seth, editor-in-chief of XML-Journal, is the chief technology officer of ikigo, Inc., a provider of XML and Web services monitoring and management software. A freelance author and well-known speaker, he regularly writes on application development using J2EE and Microsoft .NET, EAI/B2B integration, XML and Web services, and speech and wireless applications. Feel free to e-mail him with any comments or suggestions.

XML Schema Data Type	Java Data Type
xsd:string	java.lang.String
xsd:integer	java.math.BigInteger
xsd:int	int
xsd:long	long
xsd:short	short
xsd:decimal	java.math.BigDecimal
xsd:float	float
xsd:double	double
xsd:Boolean	Boolean
xsd:byte	byte
xsd:QName	javax.xml.namespace.QName
xsd:dateTime	java.util.Calendar
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]
xsd:unsignedInt	long
xsd:unsignedShort	int
xsd:unsignedByte	short
xsd:time	java.util.Calendar
xsd:date	java.util.Calendar
xsd:anySimpleType	java.lang.String

Table 1 • Data-type mapping between XML and Java

should be able to utilize different JAXB implementations by just regenerating the schema classes and doesn't require change to the actual application code.)

Availability

At the time of this writing, the JAXB specification was available in Public Draft 2 (released on Oct 4, 2002). A 1.0 beta implementation of JAXB is also available from http://java.sun.com/xml/jaxb. This public draft and 1.0 beta implementation have been used as the basis for this article. JAXB has changed considerably since it was first conceived, the primary change being that JAXB-based binding frameworks can now generate Java classes directly from XML Schemas, eliminating the need to create custom binding schema files. According to Sun's Web site, a final version of the JAXB reference implementation and specification is expected to be available by Q1/2003.

Using JAXB, Step by Step

To explore the functionality of the JAXB API, let's take a hands-on approach and look at how the API can be used to bind XML Schemas to Java objects. For starters, we'll take a simple XML Schema, Message.xsd (see Listing 1). It represents the schema of a simple application-to-application message (see Figure 2).

Executing the XML Java schema compiler (xjc) automatically creates a set of Java classes that represents the Message Schema. The JAXB implementation provides a predefined Ant task (xjc), which can be used to map XML Schemas into a corresponding set of Java classes (as shown in Ant build file [see Listing 2]; if you're new to Ant, it is one of the most popular and powerful XML-based next-generation application build toolsets). Ant can be used to compile the set of generated classes as well. Figure 3 shows a UML class diagram of the generated classes.

The resulting set of Java classes can be used by an application to interpret the XML documents representing the Message Schema (ReadMessage.java, Listing 3) or to create them (CreateMessage.java, Listing 4). As is probably clear from the Java source code, the ReadMessage application parses an input XML document and generates a simple representation of the XML Message. On the other hand, the Create Message application creates an XML-based message from the dynamic values.

JAXB Customization – Binding Declarations

One objective of the current public draft release of the JAXB specification is a seamless schema-generation capability through predefined W3C XML Schemas, which (in many cases) does not require the developer to make any changes to the structure. However, the framework does provide the flexibility for use cases that require doing so. In some scenarios, schema customization is required to resolve conflict during class generation. For example, if your input XML Schema had an element named "class", the resulting getClass() method would conflict with the already present getClass() method for java.lang.Object. To explore schema customization further, let's look at another example. Listing 5 (the corresponding schema is also shown in Figure 4) shows the usage of jxb:property and jxb:javaType tags under the lineNo element. These tags force the schema compiler to override the defaults provided by the XML Schema and use (in this scenario) the property name and the Java type specified. Similarly, the JAXB specification documents a number of such customizations that can be applied to XML Schemas to fine-tune the generation of Java classes. Schema customization enables the developer to control the generated Java classes: package names, names of derived interfaces, the names and types of methods, attribute name/type binding, and so on.

Similar to our previous example, Ant and the xjc schema compiler can be used to create the Java classes from the schema. Figure 5 shows a UML class diagram of the basic interfaces generated through the process.

Listing 6 shows how the generated schema classes can be used to perform computations on the input XML. Listing 7 shows a scenario where the Java application is manipulating the Java content tree, in this case adding a new element called "comment".

Conclusion

The development and evolution of the JAXB programming model isn't far from the issues. The JSR-31 expert group has taken quite a while to put the specification together; in fact, we are still waiting for the final specification

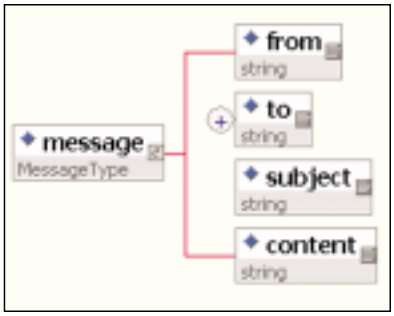


Figure 2 • Visual representation of the Message XML Schema

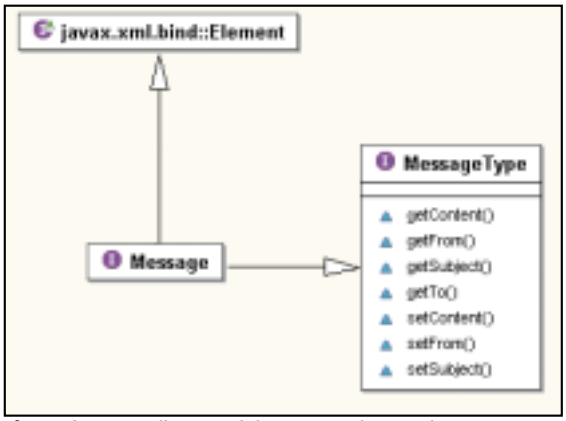


Figure 3 • UML diagram of the generated JAXB classes

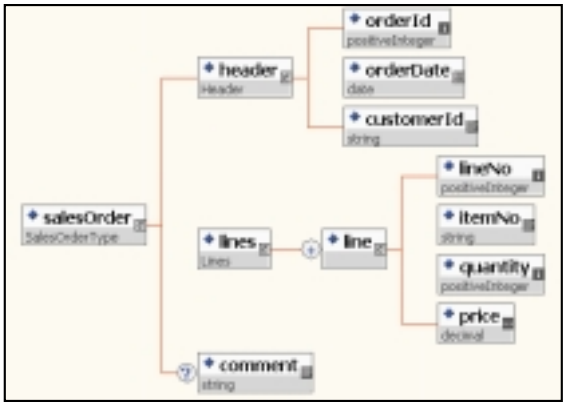


Figure 4 • Visual representation of the SalesOrder XML Schema

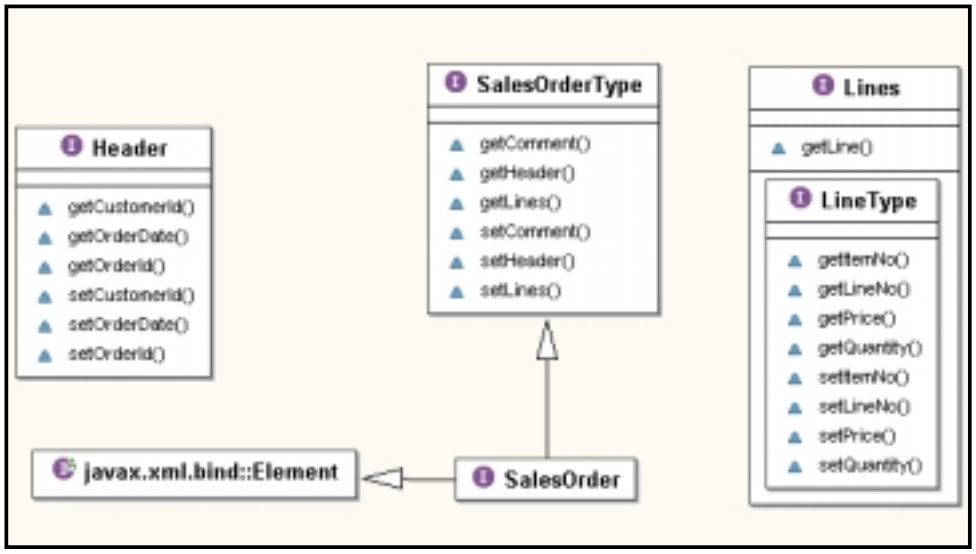


Figure 5 • UML diagram of the JAXB schema compiler–generated classes for Sales Order XML Schema

release. One probable reason for the delay of JSR-31 is the significant time that it took for XML Schema itself to be endorsed as a W3C Recommendation. Also, the current version of the JAXB specification doesn't completely map XML Schemas into Java classes. The specification identifies several scenarios as not in scope for the current version, including substitution group support, wildcard schema component, notation declarations, identity constraint definitions, and so on.

However, the ease of use that has been highlighted by the specification and reference implementation still makes JAXB a significant benefit to developers using Java and XML technologies together. Overall, we can expect that the JAXB specification/implementation will mark the beginning of standardized, automated, round-trip Java-XML binding capabilities.

As B2B interactions are automated and standardized, an efficient mechanism to generate and consume XML documents becomes more important than ever. With the emergence of Web services, this is likely to be further signified; developers will be receiving complex XML documents through SOAP-based Web services invocations. JAXB provides a much-needed Java-XML binding combination.

References

- *Java Technology for XML:* http://java.sun.com/xml
- *Java Architecture for XML Binding:* http://java.sun.com/xml/jaxb/index.html
- *JSR-31:* www.jcp.org/jsr/detail/31.jsp
- *JAXB Downloads & Specification:* http://java.sun.com/xml/downloads/jaxb.html

HITESH@SYS-CON.COM

LISTING 1 • Message.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="message" type="MessageType"/>
 <xsd:complexType name="MessageType">
  <xsd:sequence>
   <xsd:element name="from" type="xsd:string"/>
   <xsd:element name="to" minOccurs="1"
       maxOccurs="unbounded"
          type="xsd:string"/>
   <xsd:element name="subject" type="xsd:string"/>
   <xsd:element name="content" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

LISTING 2 • Ant build file build.xml

```
<?xml version="1.0"?>
<project basedir="." default="run">
    <path id="classpath">
        <fileset dir="../../../lib" includes="*.jar"
         excludes="ant.jar"/>
        <pathelement location="."/>
    </path>
    <taskdef name="xjc" classname="com.sun.tools.xjc.
     XJCTask">
        <classpath refid="classpath" />
    </taskdef>
    <target name="compile">
        <xjc schema="Message.xsd" target="."
          package="com.hks.schemas"/>
        <javac srcdir="." destdir="." debug="on">
            <classpath refid="classpath" />
        </javac>
    </target>
</project>
```

LISTING 3 • ReadMessage.java

```
import java.util.*;
import java.io.*;
import javax.xml.bind.*;
import com.hks.schemas.*;
public class ReadMessage {
public static void main(String[] args) throws Exception {
  JAXBContext jc = JAXBContext.newInstance
   ("com.hks.schemas");
  Unmarshaller u = jc.createUnmarshaller();
  u.setValidating(true);
  Message msg = (Message) u.unmarshal(new
   FileInputStream("Message.xml"));
  System.out.println("From:"+msg.getFrom());
  System.out.println("To:");
  for (Iterator iter = msg.getTo().iterator();iter.
   hasNext();) {
   String to = (String) iter.next();
   System.out.println("\t"+to);
  }
  System.out.println("Subject:"+msg.getSubject());
```

```
  System.out.println("Content:\n"+msg.getContent());
 }
}
```

LISTING 4 • CreateMessage.java

```
import java.util.*;
import java.io.*;
import javax.xml.bind.*;
import com.hks.schemas.*;
public class CreateMessage {
 public static void main(String[] args) throws Exception {
  JAXBContext jc = JAXBContext.newInstance
   ("com.hks.schemas");
  Marshaller m = jc.createMarshaller();
  Message msg = ObjectFactory.createMessage();
  msg.setFrom("hitesh@sys-con.com");
  msg.getTo().add("hitesh@sys-con.com");
  msg.getTo().add("jennifer@sys-con.com");
  msg.setSubject("Inventory Alert");
  msg.setContent("Inventory for ABC345 is down to 15%");
  m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
   Boolean.TRUE);
  m.marshal(msg,System.out);
 }
}
```

LISTING 5 • SalesOrder.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
        jxb:version="1.0">

 <xsd:element name="salesOrder" type="SalesOrderType"/>
 <xsd:complexType name="SalesOrderType">
    <xsd:annotation>
  <xsd:appinfo>
   <jxb:class name="ISalesOrder"/>
  </xsd:appinfo>
    </xsd:annotation>
  <xsd:sequence>
   <xsd:element name="header" type="Header"/>
   <xsd:element name="lines" type="Lines"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="Header">
  <xsd:sequence>
   <xsd:element name="orderId" type="xsd:positiveInteger"/>
   <xsd:element name="orderDate" type="xsd:date"/>
   <xsd:element name="customerId" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="Lines">
  <xsd:sequence>
   <xsd:element name="line" maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="lineNo" type="xsd:positiveInteger">
       <xsd:annotation>
        <xsd:appinfo>
         <jxb:property name="lineNumber"/>
         <jxb:javaType name="short"/>
        </xsd:appinfo>
       </xsd:annotation>
      </xsd:element>
      <xsd:element name="itemNo" type="xsd:string"/>
      <xsd:element name="quantity" type="xsd:
       positiveInteger"/>
      <xsd:element name="price" type="xsd:decimal"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

LISTING 6 • OrderValue.java

```
import java.util.*;
import java.io.*;
import javax.xml.bind.*;
import com.hks.schemas.*;
public class OrderValue {
 public static void main(String[] args) throws Exception {
  JAXBContext jc = JAXBContext.newInstance
   ("com.hks.schemas");
  Unmarshaller u = jc.createUnmarshaller();
   ISalesOrder order = (ISalesOrder) u.unmarshal(
     new FileInputStream("ISalesOrder.xml"));
  double amount = 0;
  for (Iterator iter = order.getLines().getLine().
   iterator();iter.hasNext();) {
   Lines.LineType line = (Lines.LineType) iter.next();
   int qty = line.getQuantity().intValue();
   double price = line.getPrice().doubleValue();
   amount = amount + (qty*price);
  }
  System.out.println("Total Order Value: $"+amount);
 }
}
```

LISTING 7 • AddComments.java

```
import java.util.*;
import java.io.*;
import javax.xml.bind.*;
import com.hks.schemas.*;
public class Main {
 public static void main(String[] args) throws Exception {
  JAXBContext jc = JAXBContext.newInstance
   ("com.hks.schemas");
  Unmarshaller u = jc.createUnmarshaller();
  u.setValidating(true);
  SalesOrder order = (SalesOrder) u.unmarshal(
     new FileInputStream("SalesOrder.xml"));
  order.setComment("Processed by JAXB!");
  Marshaller m = jc.createMarshaller();
  ...
  m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
   Boolean.TRUE);
  m.marshal(order,System.out);
 }
}
```

Download the Code
www.sys-con.com/xml

WRITTEN BY **VLADIMIR RASIN**

An In-Vehicle Human-Machine Interface Module

A new approach leveraging XML

Infotronics is a term used to describe in-vehicle multimedia, telematics, and infotainment technologies, which can be divided into a number of functional areas, such as vehicle integration, remote vehicle services, and near-vehicle interaction.

Integration involves better control and integration of a myriad of in-vehicle devices, including factory installed, dealer installed, after market, and third party. The advantage of Infotronics is the late binding of vehicle technology to take advantage of the most recent consumer/vehicle devices in a safe and secure manner. Remote services include Internet-based services commonly available to desktop computer users, such as e-mail, calendaring, e-commerce, and streaming media via cell-based network protocols such as CDMA, GSM, GPRS, and WCDMA. Near-vehicle interaction includes smart highways, tolls, gas pump–based services, and vehicle-to-vehicle safety systems such as collision detection and avoidance.

The typical vehicle design cycle takes a few years, and it's becoming increasingly difficult, if not impossible, to design an HMI (user interface) module that's flexible enough to address all needs, not only after the vehicle is delivered, but also at the time of design/production. In addition, the vehicle may have a few HMI devices operating simultaneously, further complicating the task of writing HMI applications. Taking into account the integration of consumer devices into the vehicle (as part of the convergence strategy being pursued by car manufacturers) makes everything even more complicated. Many consumer devices, such as PDAs, are HMI devices from the vehicle point of view.

With such a panoply of architectural diversity, creation of these systems is fraught with difficulty. Most vendors have chosen to simplify their solution offerings. Devices that perform these functions are sometimes called Telematics Control Units (TCU), and these solutions tend to fit in two groups.

The first group is a sort of auto personal computer – a "one box fits all" that has a graphical user interface and some reasonable voice recognition capability. This solution has radio/CD/tuner integration, climate control, e-mail, and navigation capabilities.

The second is a vehicle service provider model with a TCU that's securely connected to a remote service provider. The driver signs up for a service account that allows services (in the form of software packages) to be deployed to the TCU. These services include off-board navigation, driver assistance, e-mail, and media capabilities, to name a few.

The problem with both of these solutions is that they assume that all cars are the same. They tend to be expensive, so only the top models and brands integrate them. They become just another expensive toy. In order to be useful, the TCU has to be flexible for a wide variety of functionality and scalable in terms of price and performance for all brands and models.

Beginnings

Ford Research Laboratory built an innovative vehicle architecture that operates with all three classes of functionality. Building diverse types of vehicle applications across all the various Ford models and brands for different types of vehicle components and interactions is a daunting task – some would say impossible. This article focuses on one component of this architecture: the Human-Machine Interface subsystem, which is based on a novel XML markup language.

After looking at the user interaction part of this architecture, we decided that the "one box does all" solution was the source of too many problems. It was too expensive, not scalable across vehicle lines, too hard to upgrade, and so on. The driving costs of the one-box solution were the more expensive processor and memory needed to drive speech recognition and graphics, both of which are notorious memory and performance hogs.

A New Approach

With this in mind, we redrew the relationship with a minimum of two boxes, one for user interaction and the other to act as a hosting platform for applications. The hosting platform, since its requirements are usually less expensive, can be used in a much wider range of vehicle lines. The user interaction unit can then be differentiated by model and brand. In cheaper vehicles, a full user interface and a speech recognition system aren't required, or have fewer requirements, so this model can be made more inexpensively. In upper-end vehicles, this unit has the complete range of options available for refined comforts and tastes.

In order to take advantage of this user interaction unit, we posed the question, "How do you make this device agnostic to its intended use?" Building hundreds of customized user interfaces only displaces the cost and makes the software development life cycle very expensive. The answer: the user interface software on each unit needed to be independent of the functionally of its user interface. This sounds hard to achieve.

AUTHOR BIO

Vladimir Rasin has 16 years of experience in software and telecommunications. He worked in Russia, Israel, and Switzerland in various development and technical management positions. Prior to joining Ford, he was leading development in Professional Services department of The Fantastic Corporation (Switzerland) and before that was a software group manager at ECI Telecom (Israel). At Ford researchVladimir is responsible for the development of in-vehicle infrastructure for multimedia/telematics/infotronics applications.

One advantage of the hosting platform was that it was designed to handle complex provisioning (or deployment and installation of software). We were able to provision almost anything and decided to provision user interfaces. The hosting platform would get a new user interface and push it to the user interaction unit. The user interaction unit would then install it, and the new functionality would be present for usage.

Another design point to the system was that we wanted the hosting platform and the user interaction unit to be extremely independent of each other in terms of OS and language. While we used Java on the hosting platform, we wanted to be able to take advantage of all the user interface platform models out there. We felt that with a greater competitive landscape, Ford would not be locked into a single proprietary solution for the user interface component.

With all these design factors in play, the choice to use XML seemed self-evident. Figure 1 depicts a process we use to create in-vehicle HMI applications using XML. A user interface (graphical, speech, haptic [touch]) is created by a designer knowing little of programming. This design will be translated into XML manually or using special automated tools. It is then deployed to the vehicle and passed to the user interaction unit. The interaction unit parses the XML document and renders a representation of this user interface into a target language model (runtime only, no compilation). The user interacts with this user interface. The renderer has a shallow functionality since the majority of the service functionality (that is updateable through provisioning service) is on the hosting platform. All interaction rules are encoded.

We named this new XML markup language VUML (vehicle user interface markup language). It's currently being defined, and plans for standardization have not yet been made. The next section describes VUML life cycles and models.

Vehicle User Interface Markup Language

VUML consists roughly of two parts:

1. ***Interaction mechanism*** (between HMI and the hosting platform) described in very generic terms
2. ***Data presentation description***

The same VUML message will be used to describe the data presentation for touchscreen display or a voice-based system. The purpose of the HMI in this solution doesn't go beyond pure HMI functionality: to present the data to the user and to get input from him or her. The HMI module is unaware of the meaning of the data it's dealing with (which will be delivered using VUML). The only requirement for the HMI module in this case is that it be able to understand VUML and to implement a set of predefined user interface constructs (more on this in the "Rendering and Interaction" section).

When hosting platforms send VUML messages to the HMI module, each predefined constraint is accompanied by the name of this particular construct (naturally, there could be more than one constraint of the same type in the same HMI application). When the HMI module sends the message back to the hosting platform (as a result of the user interaction), the construct that is part of this message is again referenced by its name. This name reference is useful for the hosting platform (and HMI application running on this platform) only.

VUML has enough flexibility to describe HMI module behavior that doesn't require sending messages to the hosting platform, such as navigation between different screens.

Writing a new markup language for this purpose isn't enough; tools, deployment rules, life cycles, and rendering mechanisms are all required to satisfy the needs of the system. There are three areas associated with the architectural model. These are VUML creation, deployment, and rendering and interaction. Each area is relatively independent from the others, so independent software vendors can create commoditization in each of the areas, from VUML creation to repositories to rendering engines.

VUML creation

The choice of tools is up to the vendors. Since we're creating the first set ourselves primarily for testing, we can sketch out what a particular tool might look like.

The VUML builder is a WYSIWYG drag-and-drop editor that hides the complexity of the underlying markup detail. The design of a specific user interface starts with the selection of a specific vehicle signature, which is defined as a set of branding elements, skins, and component choices together with a set of ergonomic rules for using them. These signatures are predefined and stored in an online repository that is held by the vehicle manufacturer who's responsible for defining the signatures.

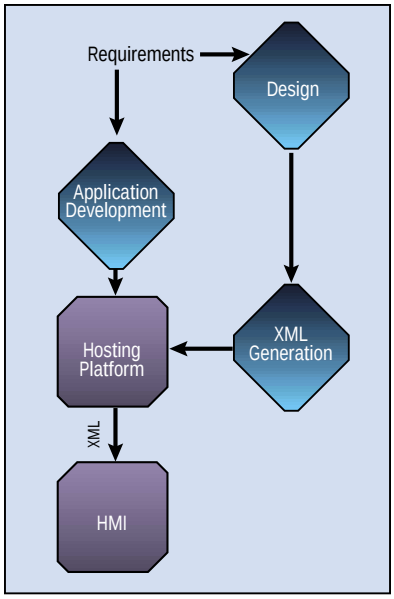


Figure 1 • Creating in-vehicle HMI applications using XML

The signature is downloaded in the builder on demand and cached for efficiency.

The builder allows the designer to describe the user interface along with actions that each component will perform. The designer can simulate the finished user interface to determine suitability and look and feel.

Once finished, the user interface can be deployed to the UI repository, where it's registered for use in a target vehicle.

Deployment

To support an efficient mechanism for user interface deployment across potentially millions of automobiles, it's necessary to define an Internet-based architecture founded on current best practices. Since this article does not focus explicitly on this part of the system, it is described in overview only.

VUML builders and deployments are accessed via a portal infrastructure (see Figure 2). Deployed user interfaces are registered for a specific automobile or group subject to security and applicability of the designer authority and vehicle type. The act of registration entails further checks on the user interface in order to check suitability and security. Once registered, the user interfaces are stored in a user interface repository. This serves two purposes: tracking and redistribution (in case the UI requires reloading).

Once registered and stored, the user interface is scheduled for provisioning to the target automobile(s). The provisioning server handles the downloading

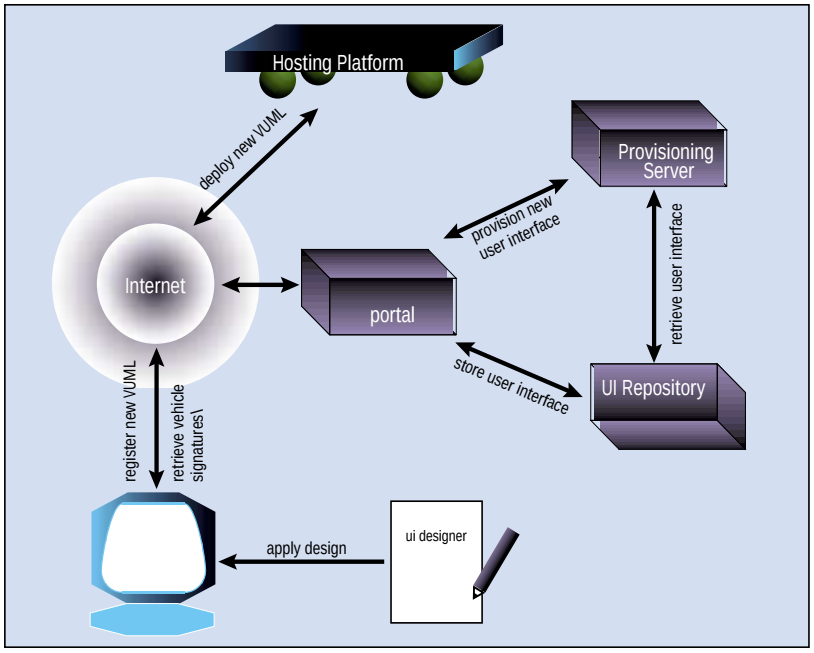


Figure 2 • Design life cycle for user interfaces

of the UI to the vehicle. Since the automobile is a mobile platform, it has all the problems associated with one. The automobile may or may not be in contact, and the type of network connectivity it has may or may not be suitable for immediate downloading. The provisioning server uses heuristics based on many factors, including connectivity parameters and the type of activity that the automobile is doing at the time. For example, deploying a new user interface while the automobile is in motion is generally a bad idea.

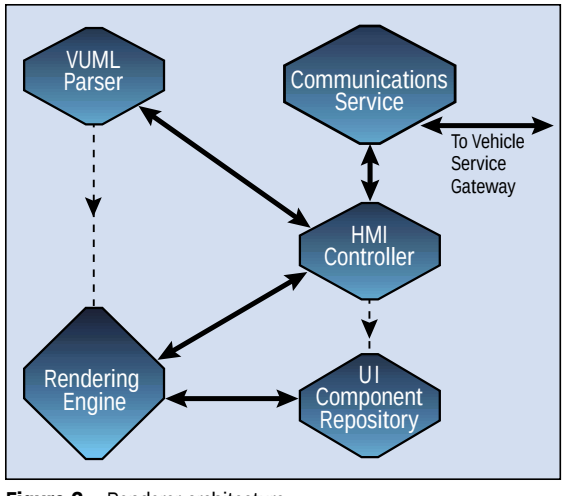


Figure 3 • Renderer architecture

In the vehicle itself, a provisioning service handles the provisioning connection to the back-end server. This provisioning server handles several different types of provisioning entities, one of which is the VUML for a new user interface. When the provisioning service receives a new user interface in VUML, it passes this to the user interaction unit (HMI). The renderer handles the parsing and translation into a new user interface.

Rendering and interaction

The rendering engine reads the VUML and creates a specific set of user interfaces using a specific language. This is done by mapping the VUML objects directly onto the language-dependent user interface constructs. For a graphical user interface, this may include panels, forms, dialog boxes, buttons, drop-down lists, etc. For a non-visual interface, it's not so straightforward. We made the decision to define a generic hierarchy of command and response categories and their respective actions for each type of user interface so that any particular HMI interaction or rendering could be described using the same constructs for different kinds of HMI modules. In other words, it will always be possible to match a particular construct to the relevant HMI module.

In a speech-based user interface, there are three components: a speech recognition engine; a speech synthesis engine; and a dictation engine, which captures speech and renders it to text form without interpreting for commands.

A haptic (touch and feedback) interface is more basic, with a set of feedback (similar in many ways to those found in games) directed to specific car areas such as shift stick, steering wheel, seat, and so on. These feedback sensory mechanisms accentuate the interaction with the vehicle, for example, collision avoidance feedback built into the steering wheel so that turning into another vehicle on your blind side results in a vibration, causing you to stop. In many ways this is more useful than an audio warning causing you to look around for the cause and increasing driver distraction. For this type of interface, VUML is targeted to those vehicle components and the actions they cause.

The specification for the HMI module defines the set of components that will be preinstalled in the HMI, although it isn't a static set since we generally provide mechanisms for loading new components into the HMI modules that support this feature. These components will be initialized (but not necessarily rendered) at startup time. The only performance hit is an initialization cost when the new VUML is loaded. Since this is done only once, it's possible to store multiple user interfaces and load them when appropriate circumstances permit, such as a diagnostic interface that automotive technicians can use for servicing.

The rendering engine is composed of multiple sections, a rendering kernel, and a UI component repository (see Figure 3). The kernel provides VUML parsing, component linking, and user interaction. The UI component repository is intended to be updateable, and the VUML allows alternative components to be used. For example, if a particular type of button, panel, or skin element were specified, the renderer would store the new component in the repository for use. The support of this dynamic mode feature depends on the underlying language target. In the case of Java, this is just another JAR file.

Summary

Ford Research Laboratory's innovative vehicle architecture, specifically the Human-Machine Interface subsystem, uses VUML, a novel XML markup language. VUML consists of an interaction mechanism and data presentation description. Associated with the architecture are creation, deployment, and rendering and interaction, which are relatively independent of one another, allowing ISVs to create commoditization in each.

VRASIN@FORD.COM

International XML Conference & Expo

XML Edge 2003

CONNECTING THE ENTERPRISE WITH XML, WEB SERVICES, JAVA, AND .NET

March 18-20, 2003
Boston, MA

XML EDGE conference&expo

web services EDGE conference&expo

JDJ EDGE conference&expo

.NET EDGE conference&expo

Featured technologies and topics will include:

- Asynchronus Messaging
- XML Standards
- SOAP
- XML Schema
- ebXML
- Validating XML Documents

Register By February 14, 2003 SAVE $400

Register By March 14 SAVE $200

Hynes Convention Center
Boston, MA

Boston
March 18-20

London
June 3-5

Berlin
June 24-26

Hong Kong
Coming soon...

For more information visit
www.sys-con.com

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver REAL XML Benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com • European & Asian Events: 011 44 208 232 1600

OWNED BY SYS-CON MEDIA
PRODUCED BY SYS-CON EVENTS

Conference program available online!
www.sys-con.com/webservices2003east

WRITTEN BY **RICHARD SALZ**

XML Within the Enterprise

Benefits, opportunities...and risk

One of XML's quieter (or at least less-hyped) success stories is its use as a tool for enterprise application integration (EAI). XML markup is being used as a common data format, and XML-based Web services are achieving success as a "protocol stack" for existing applications to integrate.

Unified EAI can provide significant cost savings that reach directly to the bottom line. One of the most notable and public examples is a leading financial services company that was able to remove 10% of their back-end servers once they moved to an all-XML messaging structure.

While this can bring economic benefits and enable new business opportunities, there are a number of risks associated with XML. As XML applications increase direct access and remote procedure calls to back-end systems, mission-critical data, and information stores – first to select partners, perhaps over a VPN, and then to a general audience over the Internet – the risks of XML become more worrisome.

We can divide these risks into two general classes: the risk of malformed XML and the risk of insecure XML.

Malformed XML

XML supports a number of different encodings of the Unicode character set. An XML document can start with a Byte Order Mark (BOM) to indicate the encoding used (see Appendix F of the XML Specification, www.w3.org/TR/REC-xml). Currently, the most common encoding is UTF-8, which isn't a BOM, and most documents therefore don't have the UTF-8 BOM.

The Expat library is an open-source XML parser originally written by James Clark. It has proven to be remarkably stable and bug-free, and has therefore become the most popular XML parser for most C/C++ applications.

Unfortunately, this stability has worked against it. The UTF-8 BOM handling was not fixed until the end of July 2001. Any XML application using a version of Expat more than 18 months old runs the risk of being crashed (there could be an uninitialized pointer value) by being sent a valid XML document that's less than a dozen characters long.

Another class issue deals with entity resolution. For example, the fragment shown in Figure 1 shows an exponential expansion that is known to crash Internet Explorer (IE).

Entities can also present additional security risks, possibly releasing internal data. For example, a simple echo service could be fooled into giving away a list of user accounts:

```
<!ENTITY foo SYSTEM "file:///etc/
 passwd">
   <echo>&foo;</echo>
```

There are also the risks of more common errors such as buffer overflows, which might have an XML-specific cause such as too many attributes, long attribute or element names, excessively nested elements (imagine a document three thousand elements deep), and so on.

Insecure XML

Unlike HTML, which is intended for display on a browser to a human, XML's use in EAI is intended for intermachine communication. As such, the risks are much greater. For example, if an attacker were to insert an extra zero while a browser is fetching an auction site's Web page, the potential bidder might notice that the price seems off by an order of magnitude and take the appropriate action.

If the document is a purchase order being sent to an online payment system, however, the automated process is much less likely to notice the modification.

While SSL can provide a tamper-proof private communication channel between two machines, it may not be sufficient. For example, if the document contains an XSLT PI, the risk of modification is now doubled: both the original

Best Practices for XML-Based Security

- **Transform all messages**
  - Isolate systems & create domains of security internally & externally
- **Sign all messages**
  - Provides authentication & prevents message tampering
- **Validate all messages**
  - Perform inbound & outbound to prevent server disruption
- **Protect against XML DoS**
  - Detect malformed, malicious XML before it reaches back-end systems
- **Encrpt message fields**
  - Conditional encryption protects without limiting business process
- **Mask internal resources**
  - Hide URLs, IP addresses, etc. with rewrites, proxies, etc.
- **Implement XML filtering**
  - Based on message size, content, or network information
- **Secure logging**
  - Audit trails for non-repudiation & to track anomalies that signal threats
- **Timestamp all messages**
  - Synchronize XML nodes, with signing, provides non-repudiation
- **Secure transport layer**
  - SSL and associated TLS

AUTHOR BIO

Richard Salz is chief security architect at DataPower Technology. Rich has been active in the Internet, distributed systems, and security sphere for over a decade. He has been a member of the IETF PKI and HTTP working groups, and is a named contributor to the HTTP/1.0 and HTTP/1.1 specs. He has also participated in various OSF, IETF, W3C, and OASIS standards projects. Rich is the Web services "endpoints" columnist for O'Reilly's XML.com Web site and is also the original author of INN, the world's most popular Usenet implementation, and ZSI, a leading SOAP implementation for the Python scripting language.

document and the stylesheet are at risk.

If trading partners are using a common schema defined by an industry consortium, it would make sense to have the schema location point to the consortium's Web site. Unfortunately, an XML application that doesn't have the ability to use a local copy is now dependent on the security of the remote site.

The XML Digital Signature specification can address these issues. Since a signature can cover multiple documents, it is possible to send a signed message that covers the main document as well as any documents (such as stylesheets or schemas) that might be referenced by the source document. Such a signature could provide a "tamper-evident seal" if the main document – or anything it explicitly or implicitly depends on – is modified.

As might be expected, this flexibility comes at the cost of implementation complexity. For example, XML doesn't care about the order in which attributes appear on an element. That is, the following are equivalent:

```
<foo a="foo" xmlns="http://www.
  example.com/xmlj">
        ...
```

and

```
<foo xmlns="http://www.example.
com/xmlj" a="foo">
    ...
```

In order for digital signatures to work, however, the XML must be output in a particular format – called canonicalization, or C14N – such that differences like the above are masked. Unfortunately, there is not a single C14N form; there are several:

- Basic C14N
- Basic C14N with comments stripped
- Exclusive C14N (for when a subset is being checked, and there are surrounding namespace declarations, such as embedding a document in a SOAP message)
- Exclusive C14N with comments stripped
- Schema-aware C14N, for when an XML Schema might add defaults, etc.
- Schema-aware C14N with comments stripped

It is quite possible for different applications to generate standards-compliant signatures, yet be unable to interoperate. For example, a UDDI-aware application might use schema-aware C14N, since that's where it was defined, but a nondirectory application (such as a simple client with a configuration file or registry settings) would probably use basic C14N.

We already have a combinatorial explosion of interoperability issues, and we haven't even talked about PKI integration, key management, and so on.

What to Do?

One approach that is gaining traction is to push XML awareness into the network. This is an evolutionary approach, building on the past successes of TCP/IP firewalls, SSL accelerators, HTTP load balancers and proxies, and so on.

A dedicated device – an XML firewall or gateway – sitting between an application and its users can provide a number of security benefits. (See Figure 2).

First, such purpose-built hardware can be immune to stack overflow attacks. If it succumbs, it should be running different operating system and hardware architecture than the application servers it's trying to protect – it makes no sense to secure an IIS Web service with an IIS-based XML firewall.

Leveraging current network technology, firewalls can provide enhanced protection against XML-based DOS attacks. For example, the firewall should perform all entity expansion, refuse to follow external links (except under strict policy configuration), validate against locally trusted schema documents, and so on.

For cryptographic use, a dedicated device can provide centralized control of private keys. Often this is more secure than placing such keys on a general-purpose application server. In addition, since an application-controlled signature is a corporate signature, not an individual signature, it makes sense to do this on a device that can be strictly controlled and configured.

Finally, a central point of control provides a single point for addressing interoperability and policy issues. Enterprise customers should make sure, however, that such configuration allows the full power and flexibility of XML to be used. As the first part of this article highlighted, XML security is first and foremost an XML processing issue.

RSALZ@DATAPOWER.COM

```
<!DOCTYPE foobar [
   <!ENTITY x0 "hello">
   <!ENTITY x1 "&x0;&x0;">
   <!ENTITY x2 "&x1;&x1;">
   ...
   <!ENTITY x100 "&x99;&x99;">
]>
<boom>
   &x100;
</boom>
```

Figure 1 • Exponential expansion that crashes IE

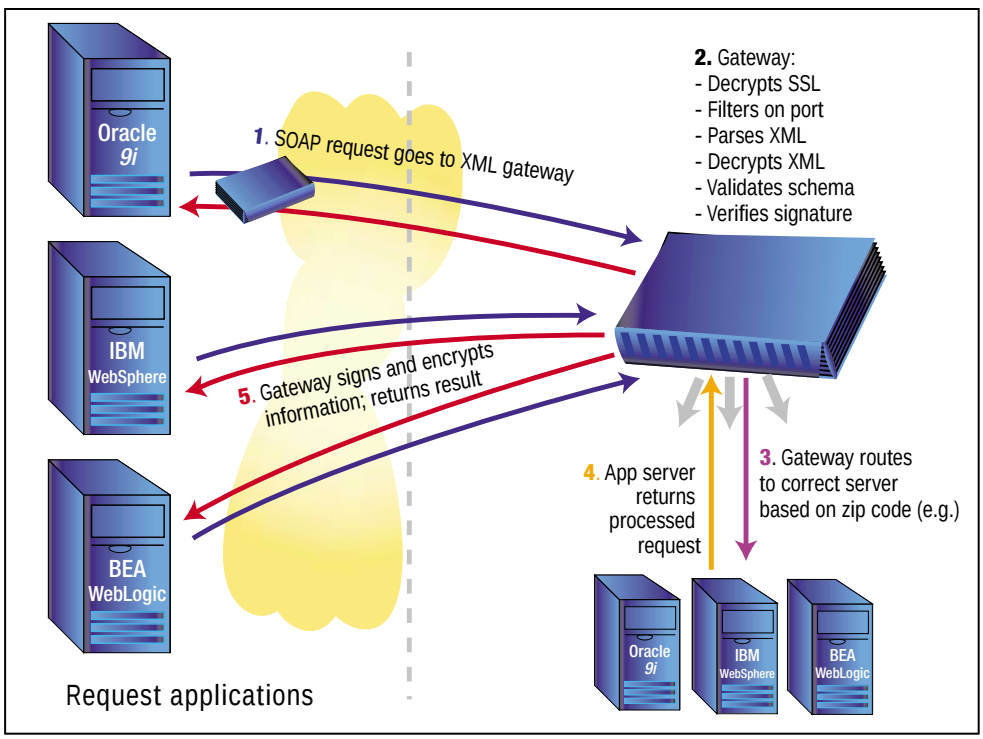


Figure 2 • Network-based XML processing

WRITTEN BY **G. KEN HOLMAN**

Real-World Use of XSL-FO

The promise of XML in printing

The XSL-FO 1.0 Recommendation is now more than a year old, and it's exciting to see a number of commercial implementations and open-source or free projects available for the Extensible Stylesheet Language Formatting Objects. One such implementation is at Crane Softwrights Ltd., where XSL-FO is used extensively on a daily basis for quality print-oriented results.

Formatting objects express the semantics of paginated formatting intent. To get professional-quality print results from XML, you transform XML instances into instances of the XML vocabulary that has been associated with the formatting semantics. Much as you'd transform your XML instances into instances of HTML for display in Web browsers, XSL-FO formatting engines interpret your intent expressed in instances of the XSL-FO XML vocabulary to produce a paginated result.

Pagination differs from the infinite-length, variable-width browser screens we've become so very used to working with. But for many developers, paginated, page-based, fixed-size-folio printable results have been left by the wayside while they focus on the HTML displays they're comfortable with. This, unfortunately, leaves a whole constituency of potential users of Web services and XML-encoded information disenfranchised. Think of those nondevelopers you know who still rely on the printed form, perhaps printing out every e-mail message or every Web screen so they can better comprehend what they're reading – they're hindered not by the content, but by the medium.

Consider also how simple requests can often result in copious amounts of information, or how many volumes of information you may have that you need to disseminate. How many people are truly satisfied with their information experience when dealing with a printed Web screen? How can you expect to properly navigate such a volume of data and ensure your users have the easiest access to the amount they have to deal with?

The semantics of pagination improve the printed result over the dynamic browser screen print function because of the nature of how the reader consumes information in these two very different environments. Navigation in the browser experience is automated; the reader of a Web screen need not know where hyperlinks to information are to be able to successfully change focus to a new location within the information set, because the browser is responsible for traversing the link. Navigation in the printed form is the responsibility of the human reading the pages. This burdens the publisher with providing navigational hints to the reader to make the reading experience easy and obstacle free. Common navigation tools used for the printed form include headers, footers, page numbers, and page-number citations. Judicious design and use of these tools by the publisher will make the message more easily accessible to the human audience.

The partner to XSL-FO is the Extensible Stylesheet Language Transformations (XSLT) Recommendation, though these two standards are surely not wedded together. Although the predominant use of XSL-FO will result from the transformation of XML information using XSLT, XSL-FO engines will work on any XML instance of the pagination vocabulary regardless of how it is generated. You can regard XSL-FO as a layout language of the printed form regardless of the programming language, database extraction, or document editing tool used. Indeed, the free AbiWord word processor has already implemented a "Save As XSL-FO" function. There is also a free resource on Crane's Web site for the dumping and display of Unicode characters and their hexadecimal code points using a Python program to directly generate an instance of XSL-FO XML for rendering by an XSL-FO engine, without any use of XSLT.

XSL-FO in the Real World

In addition to helping to serve the wonderful world of Web services to a whole new audience, XSL-FO opens doors for both the traditional publishing experience and new publishing paradigms.

Crane Softwrights Ltd. authors and publishes training material for XML technologies. Our two flagship products are "Practical Transformation Using XSLT and XPath" and "Practical Formatting Using XSL-FO." The first editions of these electronic publications predate all of the paper publications in each of these two subjects.

In our view of this new publishing paradigm, XML technologies play a critical role in making publishing efforts successful. XSLT is used to leverage a single-authored set of content into instructor-led training material for us and for our licensees around the world (creating both HTML projection slides and paper handouts), and for commercial book publishing in both electronic and paper forms. XSL-FO and XSLT are used at Crane to produce 11 electronic renditions of each book: five in a U.S. letter paper size PDF, five in an international A4 paper size PDF, and one in a hyperlinked, monospaced accessible presentation suitable for unsighted customers using on-screen text readers. Two full-size renditions of the PDF files are also richly hyperlinked, allowing successful navigation in paper form and electronically in a PDF viewer.

In this publishing scenario, the negli-

AUTHOR BIO

G. Ken Holman is the chief technology officer of Crane Softwrights Ltd., current Canadian chair of the ISO subcommittee responsible for the SGML family of standards, an invited expert to the W3C, and member of the W3C Working Group that developed XML from SGML. Ken is the author of electronically published and print-published books on XML-related technologies, and a frequent conference speaker.

gible cost of publishing new editions for existing customers allows customers to enjoy perpetual no-charge updates for future editions of the titles that they purchase. Free previews are downloadable from Crane's Web site and licenses are sold to individuals; a single geographic site license and a worldwide license for staff members of a company are available.

This leverage extends to the traditional publishing experience. Prentice Hall has purchased the print rights to each book Crane sells electronically, producing the paper-based renditions as *Definitive XSLT and XPath* and *Definitive XSL-FO*. The XML content is delivered directly from Crane to editor Dmitry Kirsanov, who polishes up the English to Prentice Hall standards, produces a print image using his own XSL-FO stylesheets, and delivers to Prentice Hall the fully composed PDF file ready for reproduction. We believe *Definitive XSLT and XPath* is the very first paper-based commercial trade book publication produced with XSL-FO.

The leverage promised by XML is realized after the first book is done and the second and subsequent books are produced. As we create new training materials, we reuse the XSLT and XSL-FO stylesheets and production environment for all of the target uses for training. We then publish the materials commercially from our Web site. If the topic is of interest in the print world, the Prentice Hall–styled stylesheets are ready for use for the document model created for our title.

XSL-FO and Our Choices

XSL-FO gives us the power to give our users information in an alternative presentation than we've used in the past with Web browsers. This increases the consumption of large amounts of information by providing well-known navigation tools consistent with other sources of information our users take advantage of.

The traditional promise of XML publishing, one source with many target formats and audiences, can be realized in the printed medium with XSL-FO, producing professional results. Consider how you can integrate traditional publishing responsibilities (user manuals, reports, etc.) with your powerful new XML information systems to produce a self-consistent multimedia collection of integrated information assets.

The new promise of Web services can reach a new audience or give an existing audience a choice in how to accept large amounts of information, giving users a better information experience and more value. Consider how you can integrate on-the-fly PDF generation as part of your service offering.

We developers may have been reluctant to address the print medium in favor of dynamic displays and may have been postponing addressing our traditional publishing requirements while we have been exploring the power XML gives us in marshaling our information. Now that printable results can be easily produced using XSL-FO, we should be considering the role that automated pagination can play in rounding out our offerings with quality print products while further leveraging our investment in markup.

Resources

- *XSL-FO 1.0 Recommendation:* www.w3.org/TR/XSL
- *XSLT:* www.w3.org/TR/xslt
- *AbiWord:* www.abisource.com
- *Crane previews and downloads:* www.CraneSoftwrights.com/xj/article-fo-real.htm
- *Definitive XSLT and XPath:* www.CraneSoftwrights.com/links/dxx-info.htm
- *Definitive XSL-FO:* www.CraneSoftWrights.com/links/dxf-info.htm
- *Dmitry Kirsanov:* www.kirsanov.com

gkholman@CraneSoftwrights.com

WRITTEN BY **JAMES A. BRANNAN**

Signing XML Using XML Signatures

A C# .NET example

XML is an accepted standard for transferring ASCII data over the Internet. Secure Sockets Layer (SSL) allows the secure transfer of documents by encrypting communication between two parties. Client certificates like X.509 certificates allow a sender's identity to be verified, i.e., when a sender initiates communication with a receiver, he or she sends a certificate that authenticates his or her identity. The sender encrypts the communication using SSL, and only the data recipient can decrypt the communication.

The W3C developed the XML Signatures specification to allow use of XML to authenticate and ensure an object's integrity – the XML signature conveniently encapsulates all the metadata associated with an object's signature. An XML signature contains a digest value generated from an object and a signature value generated from the digest value. Optionally, a signature can also contain key information on the key used to sign the digest value. Because an XML signature is an XML document, you can save the signature as a simple ASCII text document. Later, when you need to use the object, you can authenticate and ensure its integrity using the saved signature. If someone modifies the object between the time the sender signs it and the time you use it, the digest value in the signature will be incorrect and signature validation will fail. XML signatures do not replace SSL and client certificates (secure communication remains important), they build upon these technologies by allowing you to authenticate and ensure the integrity of the XML document being sent, not just the communication sending the XML.

XML signatures use Public Key Cryptography to sign objects. Public Key Cryptography is a concept in which you generate a private and a public key, which are initialization parameters to an algorithm that encrypts data. The private key encrypts data and the public key decrypts data. Keys are also useful for authenticating the sender and verifying the integrity of data encrypted with a private key. XML signatures use this ability to digitally sign XML documents.

XML Digital Signatures

Consider the XML signature shown in Listing 1. The outermost tag is the Signature tag. Signature's first child is the SignedInfo tag, shown in Listing 2.

SignedInfo contains the information to be signed in the DigestValue tag. The DigestMethod calculates the DigestValue from the original object (in Listing 1, StudentDefaults). If the underlying object is modified without recalculating the digest value, the signed object's signature is no longer valid; the digest value no longer matches the expected digest value.

The signature knows what it's signing by using the Reference tag. Reference indicates the DigestMethod used to compute the DigestValue. The CanonicalizationMethod tag indicates the canonicalization method used on the object before the DigestMethod digests the object. The SignatureMethod tag indicates the algorithm used to sign the DigestValue.

The SignedInfo tag's next sibling is the SignatureValue tag. This tag contains the DigestValue's signature:

```
<SignatureValue>EMQJkBmIYnmxD/qAX6klG
erL3hcOXgBzaJoxqEM4tHcEC0aWWbYZPRog-
EFMWAhpmXkYRzAQItcp911sY9v2AVyiPz3Xbt
gFh3UgBag/AEA8yQCIRi2uOfA/blvD94nW/Z/
7cDuMKAjCxEpPKKqzI9WY8QTJQ/27PsPM/nPX
BzIM=</SignatureValue>
```

The Object tag contains the object used to generate the DigestValue. The Reference tag, using its URI attribute, uses the Object tag's ID attribute to tie the Object tag to the SignedInfo tag. Note that in Listing 1 the signature envelops the object. An XML signature can also sign an external object by having the Reference tag point to an external URI rather than an anchor like Listing 1.

Although that sums up what you need to know about XML signatures to understand this article, you should note that I greatly simplified XML signatures and that there is much more to them. For instance, another optional tag in SignedInfo is the Transforms tag. The Transforms tag specifies any transformations that must occur to an object before calculating the DigestValue. You might wish to apply an XSLT Stylesheet to an XML document before calculating the DigestValue (without modifying the underlying XML data). Then, when the receiver validates the document, he or she knows to apply the transformation before verifying the DigestValue. To learn more on XML signatures refer to the W3C's document "XML – Signature Syntax and Processing."

Creating an XML Signature with No KeyInfo

Let's begin with a simple example that signs an XML document using a public key external to the XML signature. As you will see in the next section, you can include a public key in an XML signature using the KeyInfo tag. The code in Listing 3 applied to the XML in Listing 4 produces the XML signature in Listing 1.

AUTHOR BIO

James A. Brannan is an IT consultant in the Washington, D.C., area. He has worked extensively with XML for the past few years. He likes Java and C# equally but hates Lisp. He is currently trying to complete his masters program in software engineering at the University of Maryland, and writing occasional articles for publication.

The SignData method in Listing 3 takes a fully qualified path to an XML file as a parameter, opens the file using GetXMLFileData, signs the XML using SignXMLData, and then returns the signed XML as a string. The GetXMLFileData method opens a file containing XML and returns it as a string. The SignData method assigns the value returned by GetXMLFileData to StudentFinancialClient's strXMLData member variable. The SignXMLData method signs strXMLData.

The SignXMLData method contains the code of interest here. This code signs the XML document. The SignXMLData method creates a new XMLDocument, loads strXMLData into it, and then gets the OuterXML of the DocumentElement.

```
objDocument.LoadXml(this.strXMLData);
objDocument.LoadXml(objDocument.Docu-
mentElement.OuterXml);
```

SignXMLData gets only the DocumentElement and its children, excluding the prolog. It then creates a DataObject and assigns the XMLDocument's child nodes to the DataObject's Data member variable.

```
DataObject objDataObject = new
DataObject();
…
objDataObject.Data =
objDocument.ChildNodes;
objDataObject.Id = "StudentData";
```

The DataObject is the class that contains data to sign. The DataObject's Data member variable is an XmlNodeList. An XmlNodeList is an ordered XmlNode collection. In Listing 3, the XmlNodeList contains StudentDefault's ChildNodes (see Listing 4).

The SignXMLData method creates a key pair using the RSA class, which implements the RSA key. It then signs the XML using the private key and saves the public key to an XML file.

```
RSA objKey = RSA.Create();
string strRsaKeyXml = objKey.To
  XmlString(false);
XmlDocument objPublicDocument = new
  XmlDocument();
objPublicDocument.LoadXml
  (strRsaKeyXml);
objPublicDocument.Save("C:/
  public_key.xml");
```

To create the XML signature, SignXMLData first declares a Reference object. The Reference class corresponds directly to the XML signature's Reference tag.

```
Reference objReference = new Refer-
ence();
objReference.Uri = "#StudentData";
```

Next, SignXMLData creates a new SignedXml object and assigns the RSA key to its SigningKey member variable.

```
SignedXml objSignedXml = new
SignedXml();
objSignedXml.SigningKey = objKey;
```

SignXMLData adds the DataObject and Reference to the SignedXml and computes the signature. Computing the signature creates the digest value and signs that digest value.

```
objSignedXml.AddObject
  (objDataObject);
objSignedXml.AddReference
  (objReference);
…
objSignedXml.ComputeSignature();
```

XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ADOS Co., Ltd.	http://www.a-dos.com	81-3-5475-1551	11
Altova	www.altova.com	978-816-1600	52
BEA	dev2dev.bea.com/useworkshop		9
Borland	www.borland.com		2
CTIA Wireless 2003	www.ctiashow.com		25
Ektron	www.ektron.com/xml	603-594-0249	27
IBM	ibm.com/websphere/opentools		51
LinuxWorld	www.linuxworldexpo.com		20, 21
Macromedia	www.macromedia.com/go/cfmxad		6
PolarLake	www.polarlake.com		13
SonicXQ	www.sonicsoftware.com/websj		4
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	48, 49
XML Edge 2003	www.sys-con.com	201-802-3069	35
XMLFiles.com	www.xmlfiles.com		39
XML for Financial Services	www.worldrg.com/fw332	800-647-7600	15

Creating an XML Signature with KeyInfo

An XML signature without public key information, for example Listing 1, requires sending the public key to the signature's recipient. Another way to send the signature is by embedding the key information. Do this using the KeyInfo tag. Refer to Listing 5, a signed XML document that contains a sender's key information. The KeyInfo tag contains the key's value:

```
<KeyInfo>
<KeyValue
xmlns="http://www.w3.org/2000/09/xmld
sig#">
<RSAKeyValue>
<Modulus>8idBHzSrJuCeMT5jMnrIa/lVsQvb
CSughfOGuS9sPyKjHivqZX3kJ4/dNanmL/g1J
rYsoZloLC2Qq4lSmtTP/dw+6VdIB0YYn4XjRz
EP6S2vgg5G6Fgd5ctEEBSWtkAKPdeFhA64p2i
rkn0RCWLzVuwqtVPElMr1tN97jvb4i2k=<
 /Modulus>
<Exponent>AQAB</Exponent>
</RSAKeyValue>
</KeyValue>
</KeyInfo>
```

The code in Listing 6, run using the XML in Listing 4, produces the XML signature in Listing 5.

> "If you regularly send XML data across the Internet, and the data is important enough that you currently use SSL, and possibly client certificates, then you should consider using XML signatures"

Because Listing 6 uses an RSA key, the RSAKeyValue tag in Listing 5 contains the key's modulus and exponent. These are two values used by the RSA algorithm. Note that other key creation methods exist, for example, DSA encryption. Refer to the W3C documentation for more detail. The .NET SDK documentation contains an example using DSA encryption.

The code in Listing 6 is similar to the code in Listing 3, only it doesn't save the generated key and it adds key information to the SignedXml class using the KeyInfo class.

```
KeyInfo objKeyInfo = new KeyInfo();
objKeyInfo.AddClause(new RSAKeyVal-
ue(objKey));
objSignedXml.KeyInfo = objKeyInfo;
```

Verifying an XML Signature with No KeyInfo

Recipients wanting to check an object's signature need to use the public key of the key pair used to produce the XML signature. The code in Listing 7 checks that the XML document in Listing 1 was not modified. First, the CheckRSASignature method loads the signed XML. Note that you must be certain not to exclude the line:

```
objXmlDocument.PreseveWhiteSpace =
true;
```

If you do, then the XmlDocument will remove all white space, thus invalidating the signature. Second, it adds the XML to a SignedXml object. Third, because the code in Listing 3 did not add the key information to the XML document, the CheckRSASignature method adds the information:

```
objPublicKeyXmlDocument = new XmlDoc-
ument();
objPublicKeyXmlDocument.Load("C:/pub-
lic_key.xml");
string strPublicKeyXml = objPublicK-
eyXmlDocument.InnerXml;

RSA objRSA = RSA.Create();
objRSA.FromXmlString(strPublicK-
eyXml);

KeyInfo objKeyInfo = new KeyInfo();
objKeyInfo.AddClause(new RSAKeyVal-
ue(objRSA));
objSignedXml.KeyInfo = objKeyInfo;
```

To add the key information, CheckRSASignature loads the key into an XmlDocument and builds the key from the XMLDocument. It then adds a KeyInfo object to the SignedXml object. The CheckRSASignature method uses the SignedXml's CheckSignature method to return true (unmodified) or false (modified).

Verifying an XML Signature with KeyInfo

In Listing 5, because the XML signature embeds the key information, verifying the XML signature is easier than the signature in Listing 1. Listing 8 checks the signature for the XML in Listing 5. Listing 8 is identical to Listing 7, minus the code required to add the key information to the signature. Because the key information is part of the XML signature, when the SignedXml class loads the XML, SignedXml creates the KeyInfo object automatically.

Summary

There is much more to digital signatures than what's presented in this article. Space does not permit detailed coverage of using X.509 certificates or DSA keys, and there are more XML signature syntax variations. Note that this article uses two command-line programs and will not work in an ASP .NET page or a Web service without substantial modification. Refer to Microsoft's Knowledge Base Article 322371: PRB: "CSP for This Implementation Could Not Be Acquired CryptographicException Error During Instantiation" for more information on how to use XML signatures in an ASP .NET page or Web service.

This article should help get you started using XML signatures using C#. If you regularly send XML data across the Internet, and the data is important enough that you currently use SSL, and possibly client certificates, then you should consider using XML signatures. The C# code required to sign and check XML signatures is straightforward. Using the XML signature classes in .NET using another .NET language, like VB .NET, is also straightforward. Moreover, because XML signatures are a W3C recommendation (including a published XML Schema describing XML signature syntax), you can rest assured that an XML signature generated by a .NET application has the same syntax as an XML signature generated by a Java program.

Business transactions need trusted communication and trusted content. XML signatures are one way to ensure the authentication and integrity of an object. Because XML is a widely accepted standard in sending information over the Internet, it follows that XML signatures are the natural candidate for ensuring content's authentication and integrity.

BRANNAJ1@VERIZON.NET

LISTING 1 • Signed XML document example

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
/>
    <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#StudentData">
      <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>kD4OdqaLdEE7p6EEM0TPEMZEwOk=</Digest-
Value>
    </Reference>
  </SignedInfo>
  <SignatureValue>UPV5eNf0XNsvmw6Lb6ejyc/BGkiMeU6X1ShJIBiH-
WnYMdpr+AACTKaZ33SS
eWGn2PlhL4gcILOFA5+fXsqHIC+TcyfXbEHj//ftH3f7J+DzfhBKEwDyT07B
4ssHTSpd4jwheG/Kj8Gg94KmTuhmuCTMPZLvzYI8x2mplLgqtNuI=</Sig-
natureValue>
  <Object Id="StudentData">
    <StudentDefaults>
      <School>
        <ID>33</ID>
      </School>
      <Students submited="2">
        <Student>
          <ID>1234</ID>
          <DefaultAmount>3141.59</DefaultAmount>
          <ReportDate>2002-08-11</ReportDate>
        </Student>
        <Student>
          <ID>12341234</ID>
          <DefaultAmount>159</DefaultAmount>
          <ReportDate>1999-08-13</ReportDate>
        </Student>
      </Students>
    </StudentDefaults>
  </Object>
</Signature>
```

LISTING 2 • SignedInfo tag

```
  <SignedInfo>
    <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
/>
    <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#StudentData">
      <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>kD4OdqaLdEE7p6EEM0TPEMZEwOk=</DigestValue>
    </Reference>
  </SignedInfo>
```

LISTING 3 • Signing XML using C# - no KeyInfo

```
namespace xmlsignatures_client {
  using System;
  using System.IO;
  using System.Xml;
  using System.Security.Cryptography;
  using System.Security.Cryptography.Xml;

  public class StudentFinancialClient{
    private string strXMLData;
    public string SignData(string strDataPath){
      // get xml from file
      // sign the xml and return XML as string
      this.strXMLData = this.GetXMLFileData(strDataPath);
      return this.SignXMLData();
    }

    private string GetXMLFileData(string strFullyQualified
     Name){
      // open xml file and return as string
      string str;
      StreamReader objStreamReader = File.OpenText
       (strFullyQualifiedName);
      str = objStreamReader.ReadToEnd();
      objStreamReader.Close();
      return str;
    }

    private string SignXMLData(){
      // create document from xml string
      // create data object to sign
      XmlDocument objDocument = new XmlDocument();
      DataObject objDataObject = new DataObject();
      // load xml into the document object (getting only
      // the root node and children)
      objDocument.LoadXml(this.strXMLData);
      objDocument.LoadXml(objDocument.DocumentElement.
        OuterXml);

      // set the document's children as the data object's data
      // give the data object an id, reference

      objDataObject.Data = objDocument.ChildNodes;
      objDataObject.Id = "StudentData";
      Reference objReference = new Reference();
      objReference.Uri = "#StudentData";

      // create a RSA key and save the public key
      RSA objKey = RSA.Create();

      string strRsaKeyXml = objKey.ToXmlString(false);
      XmlDocument objPublicDocument = new XmlDocument();
      objPublicDocument.LoadXml(strRsaKeyXml);
      objPublicDocument.Save("C:/public_key.xml");

      // create a SignedXml object
      // add the key to the SignedXml
      // add the data object, the reference
      // and then compute the signature

      SignedXml objSignedXml = new SignedXml();
      objSignedXml.SigningKey = objKey;
      objSignedXml.AddObject(objDataObject);
      objSignedXml.AddReference(objReference);
      objSignedXml.ComputeSignature();
      XmlElement xmlSignature = objSignedXml.GetXml();
      // return the signed xml string
      return xmlSignature.OuterXml;
    }

    public static void Main(string[] args){
      try
      {
        StudentFinancialClient objStudentFinancialClient =
          new StudentFinancialClient();
        XmlDocument objPublicDocument = new XmlDocument();
        objPublicDocument.PreserveWhitespace = true;
        objPublicDocument.LoadXml(objStudentFinancial
          Client.SignData(args[0]));
        objPublicDocument.Save("C:/signed.xml");
      }
      catch(Exception e){Console.WriteLine(e.Message);}
    }
  }
```

LISTING 4 • Sample XML data

```
<?xml version="1.0"?>
<StudentDefaults>
  <School>
    <ID>33</ID>
  </School>
  <Students submited="2">
    <Student>
      <ID>1234</ID>
      <DefaultAmount>3141.59</DefaultAmount>
      <ReportDate>2002-08-11</ReportDate>
    </Student>
    <Student>
      <ID>12341234</ID>
      <DefaultAmount>159</DefaultAmount>
      <ReportDate>1999-08-13</ReportDate>
    </Student>
  </Students>
</StudentDefaults>
```

LISTING 5 • Signed XML Document with KeyInfo

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
```

```
    <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#StudentData">
      <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
     <DigestValue>kD4OdqaLdEE7p6EEM0TPEMZEwOk=</DigestValue>
    </Reference>
  </SignedInfo>

<SignatureValue>a96PLPpOU10kL/qOJm4cHmy/Xo+Tyb6cXm45Ajljon7i
6+rrPTDgXGYy99AHDs
LO8afNxz9bGewuaEMf7k/CX5NaxSqp7V4T4Iy1gM8XiRKJuK0s9bEr6oE0B3
3CtjrFMVKzpTfS33LYhMBVIML8aPG5yu8OMoNHni7q0WSufwM=</
  SignatureValue>
  <KeyInfo>
    <KeyValue xmlns="http://www.w3.org/2000/09/xmldsig#">
      <RSAKeyValue>

<Modulus>9J1RTwLH+BLTCmb/8hxap+PYxLOrx3EH1tMdAM/pM3zlNYrXsT+
mHyfU7hZw8
UOLLBogg45NyIOUlw8MetiJz3Nu9hl3qR8/VLbjwhF6mIl+qhxeu-
UqDymGK2VMtBpysRghiWkxLRCdcwSV5J5hyOUYg/tq3B8k+/+K/a1XkUM0=<
/Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
  <Object Id="StudentData">
    <StudentDefaults>
      <School>
        <ID>33</ID>
      </School>
      <Students submited="2">
        <Student>
          <ID>1234</ID>
          <DefaultAmount>3141.59</DefaultAmount>
          <ReportDate>2002-08-11</ReportDate>
        </Student>
        <Student>
          <ID>12341234</ID>
          <DefaultAmount>159</DefaultAmount>
          <ReportDate>1999-08-13</ReportDate>
        </Student>
      </Students>
    </StudentDefaults>
  </Object>
</Signature>
```

LISTING 6 • Signing XML using C# - with KeyInfo

```
namespace xmlsignatures_two {
  using System;
  using System.IO;
  using System.Xml;
  using System.Security.Cryptography;
  using System.Security.Cryptography.Xxml;

  public class StudentFinancialClientTwo {
    private string strXMLData;

    public string SignData(string strDataPath){
    // get xml from file
    // sign the xml and return XML as string
    // post the signed xml to the webservice, getting
       back result
    this.strXMLData = this.GetXMLFileData(strDataPath);
    string strSignedXml = this.SignXMLData();
    return strSignedXml;
    }

    private string GetXMLFileData(string strFullyQualified
     Name){
      // open xml file and return as string
      string str;
      StreamReader objStreamReader = File.OpenText
       (strFullyQualifiedName);
      str = objStreamReader.ReadToEnd();
      objStreamReader.Close();
      return str;
    }

    private string SignXMLData(){
      // create document from xml string
      // create data object to sign
      XmlDocument objDocument = new XmlDocument();
      DataObject objDataObject = new DataObject();

      // load xml into the document object (getting only
      // the root node and children)
      objDocument.LoadXml(this.strXMLData);
      objDocument.LoadXml
      (objDocument.DocumentElement.OuterXml);

      // set the document's children as the data object's
      //data
      // give the data object an id, reference

      objDataObject.Data = objDocument.ChildNodes;
      objDataObject.Id = "StudentData";
      Reference objReference = new Reference();
      objReference.Uri = "#StudentData";

      // create a RSA key and save the public key

      RSA objKey = RSA.Create();

      // create a SignedXml object
      // add the key to the SignedXml
      // add the data object, the reference
      // and then compute the signature

      SignedXml objSignedXml = new SignedXml();
      objSignedXml.SigningKey = objKey;

      objSignedXml.AddObject(objDataObject);
      objSignedXml.AddReference(objReference);

      // add the key information to the document
      KeyInfo objKeyInfo = new KeyInfo();
      objKeyInfo.AddClause(new RSAKeyValue(objKey));
      objSignedXml.KeyInfo = objKeyInfo;

      objSignedXml.ComputeSignature();
      XmlElement xmlSignature = objSignedXml.GetXml();

      // return the signed xml string

      return xmlSignature.OuterXml;
    }

    public static void Main(string[] args){
      try
      {
        StudentFinancialClientTwo objStudent
        FinancialClient = new StudentFinancialClientTwo();
          XmlDocument objPublicDocument = new Xml
           Document();
          objPublicDocument.PreserveWhitespace = true;
          objPublicDocument.LoadXml(objStudent
           FinancialClient.SignData(args[0]));
          objPublicDocument.Save("C:/signed_two.xml");
      }
      catch(Exception e){Console.WriteLine(e.Message);}
    }
  }
}
```

LISTING 7 • C# application that checks signed XML in Listing 1

```
namespace xmlsignatures_server {
  using System;
  using System.IO;
  using System.Security.Cryptography;
  using System.Security.Cryptography.Xml;
  using System.Xml;

  public class SignedXmlValidator {

    public bool CheckRSASignature(string strSignedXmlPath){

      SignedXml objSignedXml;
      XmlDocument objPublicKeyXmlDocument;

      // get the public key from file
      objPublicKeyXmlDocument = new XmlDocument();
      objPublicKeyXmlDocument.Load("C:/public_key.xml");
      string strPublicKeyXml =
        objPublicKeyXmlDocument.InnerXml;
      RSA objRSA = RSA.Create();
```

```
      objRSA.FromXmlString(strPublicKeyXml);

      // get the signed xml document and get the signed
         portion
      objSignedXml = new SignedXml();
      KeyInfo objKeyInfo = new KeyInfo();
      objKeyInfo.AddClause(new RSAKeyValue(objRSA));
      objSignedXml.KeyInfo = objKeyInfo;
      XmlDocument objXmlDocument = new XmlDocument();
      objXmlDocument.PreserveWhitespace = true;
      objXmlDocument.Load(strSignedXmlPath);
      XmlNodeList objNodeList = objXmlDocument.
        GetElementsByTagName("Signature");
      objSignedXml.LoadXml((XmlElement)objNodeList[0]);

      // check the signature to ensure no tampering
      return objSignedXml.CheckSignature();
    }

    public static void Main(string[] args){
      try
      {
        SignedXmlValidator objSignedXmlValidator = new
          SignedXmlValidator();
        Console.WriteLine(objSignedXmlValidator.
         CheckRSASignature(args[0]));
      }
      catch(Exception e){Console.WriteLine(e.Message);}
    }
  }
}
```

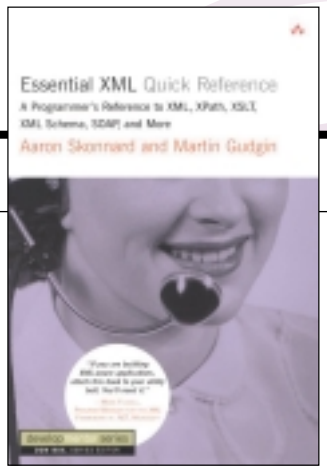
LISTING 8 • C# application that checks signed XML in Listing 5

```
namespace xmlsignatures_server {
  using System;
  using System.IO;
  using System.Security.Cryptography;
  using System.Security.Cryptography.Xml;
  using System.Xml;

  public class SignedXmlValidator {

    public bool CheckRSASignature(string strSignedXmlPath){

      SignedXml objSignedXml;

      // get the signed xml document and get the signed
      //  portion
      objSignedXml = new SignedXml();
      XmlDocument objXmlDocument = new XmlDocument();
      objXmlDocument.PreserveWhitespace = true;
      objXmlDocument.Load(strSignedXmlPath);
      XmlNodeList objNodeList = objXmlDocument.
       GetElementsByTagName("Signature");
      objSignedXml.
       LoadXml((XmlElement)objNodeList[0]);

      // check the signature to ensure no tampering
      return objSignedXml.CheckSignature();
    }

    public static void Main(string[] args){
      try
      {
        SignedXmlValidator objSignedXmlValidator = new
          SignedXmlValidator();
        Console.WriteLine(objSignedXmlValidator.
         CheckRSASignature(args[0]));
      }
      catch(Exception e){Console.WriteLine(e.Message);}
    }
  }
}
```

Download the Code
www.sys-con.com/xml

REVIEWED BY **XML-JOURNAL EDITORIAL STAFF**

Essential XML Quick Reference



First impressions

Essential XML Quick Reference

A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More
by Aaron Skonnard and Martin Gudgin

Published by: Addison-Wesley
Pages: 402
ISBN: 0201740958
List Price: $24.99

If you're a developer looking for a quick and useful reference on some of the fundamental standards around XML, look no more. In a handy paperback edition priced at $24.99, *Essential XML Quick Reference* from Addison-Wesley is a great buy. Regardless of which programming language you use, this book stands out as a great reference on the basic constructs of XML and its usage.

Organization

The book is organized into 10 chapters and contains a detailed index, which is actually quite useful given the new terms that XML and its associated standards have introduced. Each of the 10 chapters focuses on one essential aspect of XML – XML 1.0 and Namespaces, DTD, XPath, XPointer (with XInclude and XML Base), XSLT, SAX, DOM, XML Schema (covered in two chapters), and SOAP. Even though the book is really a technical reference, each chapter provides a basic introduction to its topic. I think that in the next edition of the book the authors may want to extend the topic introductions by a paragraph or two. Even though a reference book isn't really expected to teach the technology, readers would definitely benefit from an extended introduction. This is particularly appropriate for complex chapters such as the ones on SOAP and XML Schema.

My favorite is the chapter on XPath, which provides a handy reference to the expressions and the core function library. I also liked the related chapter on XSL Transformations (XSLT). The two long, detailed chapters on XML Schemas, which compose more than 30% of the book, are quite useful as well. This comprehensive coverage is quite appropriate, given that the XML Schema specification is probably one of the most complex and detailed in the set of core standards around XML. A chapter on XSL-FO would be a nice addition to the next edition to cover use of XML in print media.

Where Is myML?

As is probably clear by now, the book is all about the "core" standards around XML. What it's not about is the ever-expanding list of MLs in your favorite industry, application, etc. So if you're looking to get a perspective on the various markup languages that have been developed around human resources, manufacturing, financial services, and so on, you'll have to look elsewhere.

Code Snippets

In most places, the authors have done a good job of keeping the code snippets short and sweet to convey the message. In scenarios in which a detailed code example is required, particularly in the chapters on DOM and SAX, the authors have chosen to rely on Java and Visual Basic as the two programming languages to demonstrate the code. Being a hands-on technologist, I do like code examples, but I have to agree with the authors' style of short snippets. However, to satisfy all the C#, Perl, C++ developers, it might be a good idea to have the source code in different programming languages on a Web site. In all, I think the authors' decision to focus more on the underlying XML semantics than programming constructs is a good decision that maintains the focus of the book.

What About Web Services?

Web services, as we all know, is probably one of the most popular and growing areas in which XML is being used. Though the book doesn't focus on Web services, it does include a dedicated chapter on SOAP 1.1. My suggestion for the authors and publishers is to add a chapter on the related WSDL (Web Services Description Language) as well, which would provide a more thorough introduction to Web services. Also, I would encourage the authors and publishers to team up again and work on an *Essential Web Services Quick Reference*.

Online Version

An online version of the book (nonprintable but searchable PDF files) is available. Although I love to read books in print, the online edition is a great companion when I'm on the road.

HITESH@SYS-CON.COM

PLE MAGAZINES ONLINE

AND SAVE UP TO $400 AND RECEIVE UP TO 3 FREE CDs!

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

Pick a 3-Pack, a 6-Pack or a 9-Pack ☑

TO ORDER: Choose the Multi-Pack you want to order by checking next to it below. Check the number of years you want to order. Indicate your location by checking either U.S., Canada/Mexico or International. Then choose which magazines you want to include with your Multi-Pack order.

☐ 3-Pack	☐ 1YR ☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR ☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR ☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

☐ **Java Developer's Journal**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **XML-Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebLogic Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189 /	Save: $75 + FREE $198 CD
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **Wireless Business & Technology**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $192	You Pay: $139 /	Save: $53 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $79.99 /	Save: $16
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebSphere Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

3-Pack

Pick any 3 of our magazines and save up to $275.00

Pay only $175 for a 1 year subscription plus a FREE CD

- 2 Year – $299.00
- Can/Mex – $245.00
- International – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350.00

Pay only $395 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $669.00
- Can/Mex – $555.00
- International – $710.00

9-Pack

Pick all 9 of our magazines and save up to $400.00

Pay only $495 for a 1 year subscription plus 3 FREE CDs

- 2 Year – $839.00
- Can/Mex – $695.00
- International – $890.00

SYS-CON MEDIA

XML NEWS

DataPower, Contivo Offer Complete Standards-Based XML Integration Solution

(*Cambridge, MA*) – DataPower Technology, Inc., a provider of intelligent XML-aware network infrastructure, has announced a joint marketing and development agreement with Contivo, Inc., a provider of automated data integration, to integrate the Contivo Enterprise Integration Modeling (EIM)T solution with the DataPower XA35 XML AcceleratorT. The joint solution provides enterprise customers with a complete standards-based XML integration solution, providing unparalleled data modeling and the world's fastest XML data transformation technology to significantly reduce time and money spent on integration services.

www.datapower.com
www.contivo.com

First Public Release of Apache Forrest

Forrest is an XML standards-oriented project documentation framework based on Apache Cocoon, providing XSLT stylesheets and schemas, images, and other resources.

Forrest uses these to render the XML source content into a Web site via command-line, robot, or dynamic Web application.

One goal is to create a consistent xml.apache.org Web site, with a uniform, lightweight, easy-to-navigate layout and structure. Each project will be responsible for maintaining its own documentation and Web site, which are imported, aggregated, and published automatically by Forrest. The framework is intentionally designed to allow Forrest to also be applied in other projects.

http://xml.apache.org

Sarvega Announces Availability of Sarvega XPE 2000

(*Chicago*) – Sarvega, Inc., a provider of XML switching solutions, has announced the general availability of the Sarvega XPE 2000 Switch, the next-generation release in the award-winning Sarvega XPE product line. The Sarvega XPE 2000 is based on Sarvega's patent-pending core technology, XML EventStream Operating System (XESOSTM). The Sarvega XPE 2000 addresses the critical infrastructure challenges Global 1000 companies face when deploying XML in dynamic content presentation or in applications such as Web services.

www.sarvega.com

Open Office Standard Developed at Global Consortium

(*Boston*) – Members of the OASIS standards consortium have formed a technical committee to advance an open, XML-based file format specification for office applications. The new OASIS Open Office XML Format Technical Committee brings together representatives throughout the industry committed to establishing standard data interoperability for office applications. Their work will be suitable for documents containing text, spreadsheets, charts, and graphs and will retain high-level information for editing.

www.openoffice.org

VorteXML Server Speeds Adoption of XML

(*Lowell, MA*) – Datawatch Corporation, a leading provider of enterprise reporting, infrastructure, and IT support solutions, has released VorteXML Server, enabling users to easily automate the often laborious and lengthy process of transforming legacy data into valid XML. The product complements the VorteXML Designer, which allows users to visually extract, transform, and map data from structured text output into valid XML without programming. Together, the combined solution provides the ideal toolset to generate XML for popular e-business applications, including bill and statement presentation, B2B interactions, legacy transformation, and Web services.

www.datawatch.com

PureEdge Announces Support for XForms

(*Victoria, BC*) – PureEdge Solutions, Inc., congratulates the World Wide Web Consortium (W3C) and the XForms Working Group on its release of XForms 1.0 as a Candidate Recommendation. The next generation of PureEdge e-forms will enable XForms transactions to be signed and archived as a complete document, while passing XForms data to business processes.

"XForms standardizes the dynamic data and processing needs of complex forms and will hasten their migration from paper, and result in Web applications that are more robust, interoperable, and maintainable than is possible with HTML systems," said Dr. John Boyer, senior product architect, PureEdge Solutions.

www.uwi.com

developer.sys-con.com Just Launched!

(*Montvale, NJ*) – The newest innovation from SYS-CON Media, the world's leading i-technology publisher, is our highly interactive site for developers by developers. Check it out for its rich variety of technical meat and informed discussion threads, plus all the latest industry reporting, instant polls, and more...the whole thing is free and it's growing fast. Register at the site, and bookmark it today, but whatever you do, don't tell your senior IS/IT manager...because as new developer-oriented sites go, this one has the potential to become seriously addictive !

http://developer.sys-con.com

Attachmate myEXTRA! Smart Connectors Integrate Existing Business Processes

(*Bellevue, WA*) – Attachmate Corporation has announced the release of myEXTRA! Smart Connectors, a comprehensive line of data and logic access tools that securely expose legacy information and applications to serve new business initiatives. myEXTRA! Smart Connectors let organizations convert legacy host–based information to XML, Web services, or component technologies, such as EJB and COM+, which can be used by multiple applications across the enterprise. "Integration can account for 40% of the costs in large application rollouts," said Markus Nitschke, vice president of marketing for Attachmate. "Smart Connectors reduce the time it takes to generate new applications based on core business data from months/weeks to days/hours."

www.attachmate.com