

XML JOURNAL

The World's Leading XML Resource

Volume: 1 Issue: 2

XML-JOURNAL.COM

Announcing... **XML DevCon 2000**
Coming June 25-28, 2000
JavaCon 2000 September 24-27, 2000

FROM THE EDITOR

'I XML'... 'I XML too'
by Ajit Sagar pg. 3

XML INDUSTRY INSIDER

Introducing ebXML
by Bob Sutor pg. 16

SYS-CON RADIO

Interview with Jeremy Allaire
of Allaire Corporation, Inc. pg. 32

CORBA & XML

CORBA & XML - Hit or Myth?
by M. Elenko & D. Clarke pg. 56

DATA TRANSFORMATION

XML and Data Interchange
by Mike Hogan pg. 58

IMHO

Are You Getting Your
Company 'XML-Ready'?
by Bruce Sharpe pg. 65

**SYS-CON
MEDIA**

THE SPELL OF XML: THE NEXT INTERNET REVOLUTION

It's changing the way
people build Web sites -
and what they can
do with them

by Bruce Sharpe
see page 18

XML at Work: ColdFusion and Microsoft's XMLDOM Object

Exploring one alternative to the WDDX format

David Gassner

4

Feature: XML for B2B Integration

Now that the Web has won and the business-to-business space
is growing at an amazing rate...XML is coming into its own



Simeon Simeonov

10

XML & Databases: 'There's Too Much Confusion!'

The technology du jour may be XML, but what's needed is
to cut through the growing infoglut that's surrounding it

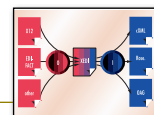


Ken North

26

Electronic Data Interchange: An Approach to Representing EDI in XML

Companies can expand their existing EDI trading systems today using XML



Jeff Ricker

36

Feature: Business to Business Integration with Trading Partner Agreements

As interenterprise integration technologies such as XML increasingly undergird
tomorrow's business models, IBM develops tpaML, an XML specification for B2B interactions...



Francis Parr

40

<e-BizML>: XML, RDBMS and OODBMS: Peaceful Coexistence?

As the database management
system wars rage on, XML comes over the horizon and makes a truce possible



Ajit Sagar

50

Soft Quad

www.softquad.com

XML**EDITORIAL ADVISORY BOARD**

COCO JAENICKE, SIMON PHIPPS, RICK ROSS, AJIT SAGAR, BOB SUTOR

EDITOR-IN-CHIEF: AJIT SAGAR
 EXECUTIVE EDITOR: M'LOU PINKHAM
 ART DIRECTOR: ALEX BOTERO
 SENIOR EDITOR: JEREMY GEELAN
 PRODUCTION EDITOR: CHERYL VAN SISE
 ASSOCIATE EDITOR: NANCY VALENTINE
 XML INDUSTRY NEWS EDITOR: ALAN WILLIAMSON
 E-BUSINESS EDITOR: ISRAEL HILERIO
 JAVA TECHNOLOGY EDITOR: JASON WESTRA

WRITERS IN THIS ISSUE

DAVID CLARKE, MARK ELENKO, DAVID GASSNER, MIKE HOGAN,
 KEN NORTH, FRANCIS PARR, JEFF RICKER, AJIT SAGAR,
 BRUCE SHARPE, SIMEON SIMEONOV, BOB SUTOR

SUBSCRIPTIONS

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
 PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

SUBSCRIPTION HOTLINE**800 513-7111**

COVER PRICE: \$8.99/ISSUE

DOMESTIC: \$49.99/YR. (6 ISSUES) CANADA/MEXICO: \$59.99/YR.

ALL OTHER COUNTRIES: \$69.99

(U.S. BANKS OR MONEY ORDERS)

PUBLISHER, PRESIDENT AND CEO: FUAT A. KIRCAALI
 VICE PRESIDENT, PRODUCTION: JIM MORGAN
 VICE PRESIDENT, MARKETING: CARMEN GONZALEZ
 CHIEF FINANCIAL OFFICER: ELI HOROWITZ
 ADVERTISING ACCOUNT MANAGERS: MEGAN RING
 ROBYN FORMA
 JDJSTORE.COM: JACLYN REDMOND
 ADVERTISING ASSISTANT: CHRISTINE RUSSELL
 ADVERTISING INTERN: MATT KREMKAU
 GRAPHIC DESIGNERS: ABRAHAM ADDO
 JASON KREMKAU
 GRAPHIC DESIGN INTERN: AARATHI VENKATARAMAN
 WEBMASTER: WEBB DIAMOND
 WEB DESIGNERS: MICHAEL MALOOF
 STEPHEN KILMURRAY
 WEB SERVICES CONSULTANT: BRUNO Y. DECAUDIN
 WEB SERVICES INTERN: BRYAN KREMKAU
 CUSTOMER SERVICE: CAROL KILDUFF
 ANN MARIE MILILLO

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.
 39 E. CENTRAL AVE., PEARL RIVER, NY 10965
 TELEPHONE: 914 735-7300 FAX: 914 735-6547
 SUBSCRIBE@SYS-CON.COM

XML-JOURNAL (ISSN# PENDING)
 is published bimonthly (6 times a year) by
 SYS-CON Publications, Inc., 39 E. Central Ave.,
 Pearl River, NY 10965-2306
 3rd Class Postage rates are paid at
 Pearl River, NY 10965 and additional mailing offices.
 POSTMASTER: Send address changes to:
 XML-JOURNAL, SYS-CON Publications, Inc.,
 39 E. Central Ave., Pearl River, NY 10965-2306.

©COPYRIGHT

Copyright © 2000 by SYS-CON Publications, Inc. All rights reserved.
 No part of this publication may be reproduced or transmitted in any form or by any
 means, electronic or mechanical, including photocopy or any information storage and
 retrieval system, without written permission. For promotional reprints, contact reprint
 coordinator. SYS-CON Publications, Inc. reserves the right to revise, republish and
 authorize its readers to use the articles submitted for publication.

WORLD DISTRIBUTION**CURTIS CIRCULATION COMPANY**

739 RIVER ROAD, NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

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

**SYS-CON
MEDIA**

WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF]

from
the
editor
in
the
editor**'I XML'... 'I XML too'**

It's always exciting to come back after a successful venture and talk about the results. **XML-Journal** hit the newsstands in March and, from all accounts, is a resounding success. This tells me two things: (1) there's a critical need for a good source of XML technology out there in the enterprise computing market today; and (2) our **SYS-CON** readers have great faith in **SYS-CON's** ability to deliver excellent quality publications for the computing and business community.

You'll find that the quality of our publication adheres to the high standards set by our sister publications such as **JDJ** and **CFDI**. In this issue, in addition to our regular columns, we have articles that focus on XML's integration with other business environments like the Web presentation layer and databases. You'll also find **XML-Js** interview with Jeremy Allaire regarding Allaire Corporation's XML initiatives.

I've received a number of e-mails about magazine circulation, subscriptions and so forth. While I'm happy to get the e-mail, for specific issues questions will be answered faster if you contact the folks at **SYS-CON** who actually do the work: subscriptions and circulation - subscribe@sys-con.com; Web site inquiries - Robert@sys-con.com; article proposals - Cheryl@sys-con.com / MPinkham@sys-con.com; advertising: Megan@sys-con.com.

One and One Makes Eleven

Buy, acquire - and to heck with the Build option. That's the name of the game today. Because of the fast time-to-market pressures, an increasing number of companies creating business applications are averse to the "Not Invented Here" syndrome (I am so happy about this). In turn, most of the vendors that play the role of application enablers are buying/acquiring functionality that they lack. Outsourcing is no longer considered a feasible option. In addition, financial analysts have a voracious appetite for acquisition moves. The market is full of new kids on the block just waiting to be bought out.

XML vendors are no exception. Since XML is a recently established technology, several vendors are coming up with new ideas (or new ways to do old things) and offering them to the general public. All this reminds me of the movie *Spartacus*. There's a memorable scene toward the end of the movie when Spartacus' defeated soldiers stand up one by one shouting, "I'm Spartacus," to prevent the Roman soldiers from identifying and crucifying Spartacus (Kirk Douglas). Similarly, the XML marketplace is rapidly being populated with vendors who want the market to know that they "do XML." I can almost hear the echoes of "I XML"... "I XML too." It's interesting that several XML vendors are falling into the "get acquired" category. Others are revamping their business with XML technologies as their main focus, instead of a byline (e.g., eXcelon Corporation). And some vendors are capitalizing on the momentum that XML technology has generated, creating new businesses that focus on a niche area, and generating extremely successful IPOs (e.g., WebMethods).

A Rose by Any Other Name Is Not...

Now it's time for my favorite soapbox. **XML-J** is designed to serve the business computing industry. This means we're not limiting our services to developers. Nor do we plan to publish content that consists of purely business issues. We'd like to do both, thus serving the wide spectrum of XML technologists. Hence, and with emphasis, I'd like to reaffirm that this magazine will continue to be called **XML-Journal**, not *XML Developer's Journal*.

Content Management

I'm still looking for more input from readers who say: "We'd like to see more coverage on such and such," or "It would be nice to see an article on thus and so." That to me would be a clear indication of how much value this magazine has for you now and can continue to have in the future. So don't be shy. Write to us. The more you interact with our team, the better the service we'll be able to provide for you.

Just a Reminder

Don't forget the biggest event of the year - **SYS-CON's XML DevCon 2000** in New York, June 25-28. We'll bring you the latest developments in this revolutionary technology as well as insights on related computing environments. Hope to see you there. ☺

**XML
DevCon
2000**

AJIT @SYS-CON.COM

AUTHOR BIO

Ajit Sagar is the founding editor and editor-in-chief of **XML-Journal** and a member of a leading e-commerce firm in Dallas, Texas, focusing on Web-based e-commerce applications and architectures. He is a leading Java and XML expert.



Exploring one alternative to the increasingly popular WDDX format

XMLatWork: ColdFusion and Microsoft's XMLDOM Object

XML is experiencing explosive growth as a means of transmitting data between corporate computer systems. But while one language- and platform-neutral technology for XML data mapping, the Web Distributed Data Exchange (WDDX) format, has met with an enthusiastic response, among ColdFusion developers in particular, the rest of the computer world hasn't exactly been beating a path to the WDDX door. So what are the alternatives?

When working out data-sharing systems with an organization's business partners, you as a developer may be asked to send and receive data in a more conventional XML format. This may be because those partners prefer to speak more generic forms of the XML language. If so, there are a number of tools available to developers in this situation.

If you're using ColdFusion 4.5, for example, you can create an XSL file (eXtensible Stylesheet Language) that describes to an XML Parser how to transform the XML file to a format of your choosing. The Allaire Corporation has published a Tech Note that describes this method in some detail (see the "XML Resources" list at the end of this article).

In this article I'm going to describe an alternative approach: you can parse the XML file using a series of calls to Microsoft's XMLDOM parser object and selectively copy the data to ColdFusion variables one piece at a time. I'll show how to use the XMLDOM COM object to do the heavy lifting and how to move the data from your business partner's XML file to a custom-designed ColdFusion complex data structure.

I'm using a sample XML file provided by iSyndicate, a company that provides

syndicated news feeds to thousands of Web sites (see Listing 1). The ColdFusion page with this article (see Listing 2) loads the XML document using the MS XMLDOM object, then parses the XML file with a few ColdFusion commands. I'll show how to read and transform iSyndicate's XML-based data into a complex multilevel ColdFusion structure. Then I'll show one way to use the retrieved information.

XML Structure

An XML document consists of a series of hierarchical nodes, each of which can be seen by ColdFusion through the parser object. The top level of an XML tree is called the *document*; it contains all of the XML file's data. The document contains multiple nodes, each representing a data entity. Each node can have multiple attributes, each representing a value associated with that data entity. A node can also be parent to another set of child nodes, creating one-to-many relationships.

In iSyndicate's sample file, each of the top set of nodes represents a channel – a set of news headlines collected by subject matter. Each channel node has these attributes:

- *cid*: The channel's unique identifier
- *title*: The channel's descriptive name

Each channel in turn contains multiple headline nodes. Each headline represents a news article; it contains a text value (more on this later) and these attributes:

- *href*: The URL where a reader can view the entire story
- *date*: Date of the headline

XMLDOM Reference

Microsoft's XMLDOM COM object has extensive documentation and the ability to both parse and create well-formed XML files. Parsing a tree like the one provided, though, requires only a small subset of the interface. Here's a quick reference to the objects, methods and properties I'll use:

XMLDOMDOCUMENT

The top node containing the entire XMLDOM tree

- *Async*: When set to True (the default setting), returns control to the caller before the download is finished. In ColdFusion this can cause a partial document load, so we set it to False.
- *Load(url)*: Loads the XML tree from a Web document.
- *getElementsByTagName(nodename)*: Returns a list of document-level nodes where the NodeName matches the argument.
- *hasChildNodes()*: Returns True if the node has child nodes.

XMLDOMNODELIST

A collection of nodes in the document tree, supporting iteration with <cfloop>.

XMLDOMNODE

A single node in the document tree.

- **hasChildNodes()**: Returns True if the node has children, False if it doesn't
- **childNodes()**: Returns all of the node's child nodes as an XMLDOMNodeList
- **selectNodes(nodename)**: Returns an XMLDOMNodeList of the current node's children where the NodeName matches the argument
- **Attributes**: Returns a collection of attributes, each of which consists of a key-value pair
- **Text**: The value between the node's begin and end tags

Install the COM Object

Make sure you've first installed the XMLDOM object on your server before you try running the parsing routine. The COM object is available for download at <http://msdn.microsoft.com/downloads/tools/xmlparser/xmlparser.asp>.

To install the COM object, run the self-extracting executable XMLRedist.exe file and follow the prompts.

STEP 1: INSTANTIATE THE COM OBJECT AND LOAD THE XML TREE FROM THE DATASOURCE URL

The COM object is instantiated with <CFOBJECT> (see Listing 2), and then the XML file is read and interpreted by the object using the document's Load() method. The Load() method takes a single argument: the location of the XML file as a URL. If the file is empty, or if the file isn't found at all, the COM object doesn't throw an error; instead, it returns an empty XML tree. Check for this condition before proceeding any further, using the document's hasChildNodes() method.

STEP 2: RETRIEVE THE DOCUMENT'S TOP NODES

This XML document's top nodes are known as channels. So I retrieve the nodes with the document's getElementsByTagName(nodename) method. I create a structure to hold the channels and then iterate through each channel to collect its data:

```
<cfset channel s = xml Doc. getEl e-
mentsByTagName("channel s")>
<cfset structChannels = structnew()>
<cfloop col lecti on="#channel s#"
item="thi sChannel ">
...collect the data...
</cfloop>
```

STEP 3: COLLECT EACH NODE'S DATA

A node can have both attributes and child nodes. Collecting a node's attributes requires two steps:

1. Place the attributes in a collection.
2. Loop through the collection to identify and collect the attributes you're interested in.

For each channel, I'm interested in the title (the channel's descriptive name) and the channel ID (known in the XML structure by the NodeName "cid"). I use <cfloop> to go through the channel's attributes one at a time and collect only those attributes I want.

Once the attributes have been collected, I can create keys for the channel structure based on the channel's unique ID. I also create an array to store headlines information as the final key in the structure:

```
<cfset structChannel s[thi sChannel ID]
= structNew()>
<cfset structChannel s[thi sChan-
nel ID]["ti tle"] = thi sTi tle>
<cfset structChannel s[thi sChan-
nel ID]["headl i nes"] = arraynew(1)>
```

STEP 4: COLLECT THE CHANNEL'S HEADLINES

The channel's headlines are child nodes of the channel. I use the method selectNodes(), which returns a new collection of nodes. SelectNodes operates with the same syntax as the document method getElementsByTagName(), but it only returns children of the current node.

For my purposes, the important attributes are the href and the date. Notice that the headline text isn't an attribute itself, but rather is placed between the headline's begin and end

tags. Any value in this location is the node's text property.

Now I retrieve and loop through the node's attributes, and assign the values to keys in the headline's structure. I also assign a key to the text property. And the entire structure gets appended to the current channel's array of headlines:

```
ArrayAppend(structChannel s[thi sChan-
nel ID]["headl i nes"], structnew());
newHeadl i nel D = ArrayLen(structChan-
nel s[thi sChannel ID]["headl i nes"]);
structChannel s[thi sChannel ID]["head-
l i nes"][newHeadl i nel D]["href"] =
thi sURL;
structChannel s[thi sChannel ID]["head-
l i nes"][newHeadl i nel D]["date"] =
thi sDate;
structChannel s[thi sChannel ID]["head-
l i nes"][newHeadl i nel D]["text"] =
thi sHeadl i ne. text;
```

Results

Once the entire XML tree has been retrieved, I end up with a structure of channels. Each channel has a structure of attributes, including an array of headlines, and each headline has a structure of attributes. Along the way, I've been using <cfoutput> to write my data to the returned HTML page, so I can see the results of the XML parsing.

The document loading and parsing tasks can take some time (on my slower development system it can take up to 1.5 seconds for each run through the code), so I don't recommend using the XMLDOM object in the background for each new page called by your users. Instead, try setting up scheduled tasks that refresh the data periodically. For a news

Your News Feed

Internet Wire: Small Business News

- [Microsoft Announces 1999 Industry Solution Award Winners](#)
Thu, October 28, 1999 12:42 PM
- [Hewlett-Packard Company and HNC Retek Achieve New Level of Performance](#)
Thu, October 28, 1999 12:42 PM
- [Joint Venture Unveils Smart Permit System](#)
Thu, October 28, 1999 12:42 PM

CBS MarketWatch: News

- [Infoseek loses less than expected](#)
Thu, October 28, 1999 04:20 PM
- [Amerindo distributes gains](#)
Thu, October 28, 1999 03:28 PM
- [Autobytel.com exceeds expectations](#)
Thu, October 28, 1999 03:28 PM

FIGURE 1 Dynamic page creation from cached news feed data

AUTHOR BIO

David Gassner is the owner of Vintage Business Applications, a custom software development company located in Napa, California, specializing in intranet business applications. An Allaire certified instructor, he is author of the soon-to-be released CFProject, a ColdFusion-based project management tool for workgroups.

feed, every 15 minutes is about as frequent a refresh rate as you would ever need. And since the entire dataset is now stored in a single multilevel structure, there are plenty of options regarding how to use the information.

I wanted to be able to deliver customized news feeds, allowing each user to decide what channels they wanted to see. I didn't need to search by headline, so I stored the entire structure in the application scope:

```
<cflock name="#application.appli-
cationname#" timeout="30" type="excl-
usive">
<cfset application.channels =
structChannel s>
</cflock>
```

Each user was assigned a comma-delimited list of channels that was stored in the session scope. Displaying the information was a matter of looping through the list of their assigned channels and retrieving and displaying the appropriate feeds (see Listing 3). Since the news feed data was always cached in memory, the dynamic page creation (see Figure 1) was very fast, making for a scalable solution. (When trying this code, you'll need to include a <CFAPPLICATION> tag in your application.cfm file to activate application variables.)

Conclusion

We all hope for the day when the rest of the Web world sees the light and

moves to WDDX. Until that time, however, it's comforting to know that we can easily retrieve and use conventional XML-based data with tools you can leverage in ColdFusion. ☺

XML Resources

- *Allaire's Tech Note on XML/XSL Data Transformation:*
www.allaire.com/handlers/index.cfm?ID=14244&Method=Full#1020244
- *Microsoft's Documentation on the XMLDOM Object:*
<http://msdn.microsoft.com/library/pssdk/xmlsdk/xmlid2r11.htm>

DGASSNER@NAPANET.NET

LISTING 1

```
<?xml version="1.0"?>
<newsfeed>

<channel
cid="internet_wire_small_business_hl"
title="Internet Wire: Small Business News">

<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%2526pbbx%252581%25257N%25257Neee.qvmzvmbeqzm.
kwu%25257Nbmkpvmeea%25257Naj%25257Naj 445791.l at"
date="Thu, October 28, 1999 12: 42 PM"
time="12: 42 PM PST">Microsoft Announces 1999
Industry Solution Award Winners</headline>

<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%2526pbbx%252581%25257N%25257Neee.qvmzvmbeqzm.
kwu%25257Nbmkpvmeea%25257Naj%25257Naj 445799.l at"
date="Thu, October 28, 1999 12: 42 PM">
Hewlett-Packard Company and HNC Retek Achieve
New Level of Performance</headline>

<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%2526pbbx%252581%25257N%25257Neee.qvmzvmbeqzm.
kwu%25257Nbmkpvmeea%25257Naj%25257Naj 445798.l at"
date="Thu, October 28, 1999 12: 42 PM">
Joint Venture Unveils Smart
Permit System</headline>

</channel>

<channel cid="cbs_marketwatch_news_hl"
title="CBS MarketWatch: News">

<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%2526pbbx%252581%25257N%25257Nkj a.ui zsmbei bkp.
kwu%25257Ni zkpqdm%25257N64446573%25257Nvmea%
25257Nkczzmvb%25257Nnamms.pbf%25258Nawczkm%
25258Lj ty%25257Nqagvl %2525711 qab%25258Lqagvl "
date="Thu, October 28, 1999 04: 20 PM"
time="04: 20 PM PST">Infoseek loses
```

less than expected</headline>

```
<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%252581%25257N%25257Nkj a.ui zsmbei bkp. kwu%25257Ni z
kpqdm%25257N64446573%25257Nvmea%25257Nkczzmvb%252
57Ni umzqvl w.pbf%25258Nawczkm%25258Lj ty%25257Nqagv
l %2525711 qab%25258Lqagvl "
date="Thu, October 28, 1999 03: 28 PM">
Amerindo distri butes gains</headline>
```

```
<headline
href="http://headlines.syndicate.com/pscripts/
hit/UZRinnm%2526qvmzmb-eqzm.aui tt_jcaqvmaa.pt
%252581%25257N%25257Nkj a.ui zsmbei bkp. kwu%25257Ni z
kpqdm%25257N64446573%25257Nvmea%25257Nkczzmvb%252
57Ni j bt.pbf%25258Nawczkm%25258Lj ty%25257Nqagvl %25
25711 qab%25258Lqagvl "
date="Thu, October 28, 1999 03: 28 PM">
Autobytel .com exceeds expectations</headline>
```

```
</channel>
</newsfeed>
```

LISTING 2

```
<!-- xml ParseNewsFeeds.cfm -->
<!-- Author: David Gassner -->

<cfsetting enablecfoutputonly="Yes">

<cfobject type="COM"
name="xml Doc"
class="Microsoft.XMLDOM"
action="CREATE">

<cfset xml Doc.async = false>
<cfset xml Doc.load("http://l ocal host/I Syndicate.xml")>

<cfif not xml Doc.hasChildNodes()>
<cfoutput><b>Error: Empty XML Tree</b></cfoutput>
<cfabort>
</cfif>

<!-- Collect the document's channels and
create a structure to hold them -->
<cfset channel s = xml Doc.getElementsByTagName("channel")>
<cfset structChannel s = structnew()>
```

Info Shark

www.infoshark.com

```

<!-- Loop through the channels -->
<cfl loop collection="#channel s#" item="thisChannel">

    <!-- Collect the current channel's attributes -->
    <cfset channelAttributes = thisChannel.attributes>
    <cfl loop collection="#channelAttributes#" item="thisAttribute">
    <cfswitch expression="#thisAttribute.nodename#">
    <cfcase value="title">
        <cfset thisTitle = thisAttribute.text>
    </cfcase>
    <cfcase value="cid">
        <cfset thisChannelID = thisAttribute.text>
    </cfcase>
    </cfswitch>
    </cfl loop>

    <!-- Save the channel info to the structChannels structure -->
    <cfset structChannels[thisChannelID] = structNew()>
    <cfset structChannels[thisChannelID]["title"] = thisTitle>
    <cfset structChannels[thisChannelID]["headlines"] =
arraynew(1)>

    <!-- Display the Channel as a header -->

<cfoutput><b>#structChannels[thisChannelID]["title"]#</b></cfoutput>

    <!-- Start a new unordered list -->
    <cfoutput><ul></cfoutput>

    <!-- Retrieve and loop through the channel's headlines-->
    <CFSET headlines = thisChannel.selectNodes("headline")>
    <cfl loop collection="#headlines#" item="thisHeadline">

    <!-- Retrieve and loop through the current Headline's
attributes -->
    <cfset headline_attributes = thisHeadline.attributes>
    <cfl loop collection="#headline_attributes#" item="thisAttribute">
        <cfswitch expression = "#thisAttribute.nodename#">
        <cfcase value="href">
            <cfset thisURL = "#thisAttribute.text#">
        </cfcase>
        <cfcase value="date">
            <cfset thisDate = "#thisAttribute.text#">
        </cfcase>
        </cfswitch>
    </cfl loop>
    <!-- end Headline attributes
analysis -->

    <!-- Save the headline to the headlines array for the current
channel -->
    <cfscript>
        ArrayAppend(structChannels[thisChannelID]["headlines"],
structnew());
        newHeadlineID =
ArrayLen(structChannels[thisChannelID]["headlines"]);
        structChannels[thisChannelID]["headlines"][newHead-
lineID]["href"] = thisURL;
        structChannels[thisChannelID]["headlines"][newHead-
lineID]["date"] = thisDate;
        structChannels[thisChannelID]["headlines"][newHead-
lineID]["text"] = thisHeadline.text;
    </cfscript>

    <!-- Output the headline -->
    <cfoutput>
    <li><a href="#thisURL#">#thisHeadline.text#</a>
    </cfoutput>

```

```

</cfl loop>
<!-- End headlines loop -->

<!-- End the unordered list -->
<cfoutput></ul></cfoutput>

</cfl loop> <!-- end of channels loop -->

<!-- Save the Channels to an application scope -->
<cfl lock name="#application.applicationname#" timeout="30"
type="exclusive">
    <cfset application.channels = structChannels>
</cfl lock>

```

LISTING 3

```

<!-- xmlDisplayNewsFeeds.cfm -->
<!-- Author: David Gassner -->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>
<head>
    <title>Your News Feed</title>
</head>

<body>
<h3>Your News Feed</h3>

<cfif not isdefined("application.channels")>
    <b>News Feeds aren't currently available!</b>
</body></html>
<cfabort>
</cfif>

<!-- Set a local list; this could also be a session or
client variable -->
<cfset mychannels="internet_wire.small_business.hl,cbs_market-
watch.news.hl">

<!-- Loop through the selected channels and
grab them one at a time as a local structure -->
<cfl loop list="#mychannels#" index="cid">
    <cfl lock name="#application.applicationname#" timeout="30"
type="readonly">
        <cfset channel=application.channels[cid]>
    </cfl lock>

    <!-- For each channel, display the Channel Title, then
each Headline -->
    <cfoutput>
        <ul><b>#channel.title#</b>
        <cfl loop from="1" to="#ArrayLen(channel.headlines)#"
index="iHeadline">
            <li>
                <a href="#channel.headlines[iHeadline].href#"
target="newwindow">#channel.headlines[iHeadline].text#</a><br>
                <small>#channel.headlines[iHeadline].date#</small>
            </cfl loop>
        </ul>
    </cfoutput>
    <!-- End of channel loop -->

</body>
</html>
xz

```



Sequoia

www.xmlindex.com



XML FEATURE

XML FOR B2B INTEGRATION

Now that the Web
has won and the
business-to-business
space is growing at
an amazing rate...

XML is coming
into its own

It wasn't long ago that computer industry pundits still thought that COM and CORBA would become the Internet business-to-business (B2B) integration infrastructure. Yet nowadays B2B integration on the Internet is done using XML on top of simple protocols such as HTTP, FTP and SMTP. The Web has won because of its simplicity, ubiquity and heterogeneity. XML has become the lingua franca of B2B because of its inherent capabilities: simplicity, extensibility and ease of processing.

In this article I'm going to discuss the role XML has to play in this business-to-business space, particularly with respect to the infrastructure technologies that enable B2B integration. I'll present three complementary views of B2B integration, each creating different requirements for XML use, then focus on two principal technology areas – XML data mapping and schema translation – that are common to all three views. Bear in mind that the B2B space is very broad and that every e-business-focused company has some B2B initiatives. Hopefully, this article can serve as a good starting point for your own foray into this exciting area.

Views of B2B Integration

There are three main views of B2B integration:

- The business process view
- The syndication view
- The functional view

THE BUSINESS PROCESS VIEW

The business process view focuses on the types of business processes that are being integrated, the process-specific information that needs to be exchanged and the business rules associated with the interaction.

Consider an example business-to-customer (B2C) interaction in which a book is bought on Amazon.com. B2B happens behind the scenes because amazon.com will outsource the delivery of the book to FedEx. The order fulfillment process at Amazon will have to integrate with FedEx's order-taking process. In the case of door-to-door tracking, Amazon's order tracking system will have to integrate with the package tracking system at FedEx. The basic information that has to be exchanged is the customer address information and the tracking number. XML efforts in this area focus on establishing standard XML schemas for representing information pertaining to business process integration.

THE SYNDICATION VIEW

The syndication process is often applied as another model for analyzing B2B integration. At its basic level, syndication on the Internet has to do with the creation, aggregation, distribution and consumption of some electronic asset. For example, Jim Davis is the creator of Garfield cartoons. The cartoons are aggregated by Uclick.com, a company related to Universal Press Syndicate. Uclick distributes the cartoon to companies such as AOL and New York Times, Inc. People everywhere view (consume) the cartoons. There are many forms of syndication on the Net. The syndication view of B2B focuses on analyzing what happens at every stage of the process.

There are many XML standards currently in development that address aspects of the syndication process. Notable among these is the Information & Content Exchange (ICE) protocol. ICE facilitates the exchange and management of electronic assets between members of a

syndication network. It defines the rules and mechanisms by which the assets are exchanged. An asset can be anything represented in XML. Unfortunately, the relative complexity of ICE and the high cost of solutions based on it have limited its market reach.

It's useful to think of what's being syndicated in three broad categories: content, data and services. Content can be plaintext, HTML, PDF or some XML describing publishable materials. Content is produced at least partially by humans and is generally meant for human consumption, not for machine processing. Garfield's cartoon is a good example of syndicated content. Arbitrary binary content (such as images) is usually represented as base64-encoded XML content.

Syndicated data is represented in an XML format that is machine-generated and meant for machine consumption. The XML maps to some application-level data structures, be they the results of a database query, the data of an enterprise business object or the parameters to an enterprise resource planning (ERP) system operation. A good example of syndicated data would be the information about unique customer visits to your hosted Web site. Your ISP provides the data. Ideally, it will come in not as a PDF report but in an easy-to-process XML format so that, for example, your customer profiling server can easily analyze the data.

At a deep philosophical level the distinction between content and data disappears, but for our purposes it's an important one. We have to worry about the way in which application-level data structures are converted to and from XML. I'll take a further look at this in the next section.

Syndication of services is very different from content and data syndication in that there isn't any single piece of content (article, picture, piece of data) that's being exchanged. To syndicate a service means to provide the right to access some remote functionality via a well-defined public API. For example, consider FedEx exposing access of its package tracking system to amazon.com's order tracking system. There's no specific content or data being syndicated. Instead, when customers look at the status of their book orders, Amazon has to take the tracking number that FedEx provided when they shipped the book and make a request against the FedEx system. The request will be in XML. The response will be some XML specifying the delivery status of the book. It is the request-response nature of processing that distinguishes the syndication of services. By comparison, content and data syndication constitutes a one-way dump of information.

THE FUNCTIONAL VIEW

The third and final view of B2B integration that we have to consider focuses on the types of functionality a B2B system must offer.

The major areas of functionality are defined by the operations that need to be performed in the business process and syndication views. For data syndication, we need to be able to map data to and from XML. For both content and data syndication, we need some form of robust business quality messaging to get the XML from one point to another. For the syndication of services, we need some mechanism to define XML-based remote APIs, specify/send requests and receive/process responses. Last but not least, for interoperability between information formats for business processes, we need some XML data conversion capabilities.

XML messaging has already been discussed to some extent in Sandeep Nayak's article in the premier issue of *XML-Journal* ("XML Middleware," Vol. 1, issue 1). All current technologies for XML-based remote APIs rely on XML data mapping technologies. Accordingly, the rest of this article will focus on XML data mapping and conversion technologies.

XML Data Mapping

XML data mapping has to do with generating XML from application data and creating application data from XML.

Application data covers the common datatypes developers work with every day: boolean/logical values, numbers, strings, date-time values, arrays, associative arrays (dictionaries, maps, hash tables), database

recordsets and complex object types. The process of converting application data to XML is called *serialization*. The XML is a serialized representation of the application data. The process of generating application data from XML is called *deserialization*.

The traditional approach for generating XML from application data has been to sit down and custom-code how data values become elements, attributes and element content. The traditional approach of working with XML to produce application data has been to parse it using a simple API for XML (SAX) or Document Object Model (DOM) parser. Data structures are built from the SAX events or the DOM tree using custom code. There are, however, better ways to map data to and from XML using technologies specifically built for serializing and deserializing data.

XML data mapping technologies have been developed in a variety of contexts. In most cases, XML data mapping is treated as a service within larger B2B XML technologies, e.g., messaging or request/response handling. (See the "XML References" at the end of this article, especially: Lightweight Distributed Objects (LDO), Schema for Object-Oriented XML (SOX), Simple Object Access Protocol (SOAP) and XML Schema Part 2: Datatypes.) One example technology can illustrate what XML data mapping is all about.

Web Distributed Data Exchange (WDDX) is a language- and platform-neutral technology for XML data mapping. I have to come clean and confess that I created WDDX back in 1998, so my views may be somewhat partial. But the fact of the matter is that WDDX is used by tens of thousands of server installations on the Internet. It comes as part of two Web application servers: Allaire ColdFusion and PHP. WDDX also supports Java, JavaScript, Perl, Python, ASP and COM. It is a free, open source technology managed by wddx.org.

WDDX lets developers achieve XML data mapping (a) without their knowing any XML, and (b) without their having to write any custom

code for data conversion. When you use WDDX, you don't have to worry about XML at all. With one line of application code you can convert your data to XML and with one line of application code you can get data back from the XML.

WDDX does its job by defining a single XML format for representing data. The format is specified by the WDDX DTD. Therefore the details of what format of XML to use are all taken care of. In addition, every language and platform that supports WDDX defines two platform-specific modules (see Figure 1). The serializer module converts data from that programming language to a chunk of XML conforming to the WDDX DTD. The deserializer module takes WDDX, parses it and creates data structures in the specific programming language. For example, the serializer/deserializer modules for JavaScript deal with JavaScript arrays while the ones for Java work with java.util.Vector objects. In your projects you simply use the serializer/deserializer modules that come as part of the WDDX Software Development Toolkit (WDDX SDK). The SDK is distributed by wddx.org.

WDDX is perfect for data syndication and remote B2B integration APIs because it's all about representing data as XML. For example, Moreover.com, the Web feed company, exposes all its content through a WDDX-based remote API. Access <http://moreover.com/cgi-local/page?index+wddx> with an XML-aware browser such as IE 5.0 and you'll get a WDDX packet with current headline news. A simplified version of the packet is shown in Listing 1. We can see from it that the data format is a recordset (tabular data) with three fields containing the URL to the full article, its headline text and the publishing source.

WDDX is flexible enough to handle most useful datatypes, but it doesn't give you control over the generated XML. However, there are some cases where a specific XML format is required and you need to be able to map from that specific XML format to a reasonable data structure representation of it. In this case you focus on the XML and don't care so much about the data structures. Enter schema compilation tools.

The term *XML schema* is broadly used to refer to any type of XML format specification, e.g., DTDs, XML Schema, XML Data, and so on. Schema compilers are tools that analyze XML schema and code-generate serialization and deserialization modules specific to the schema (see Figure 2). These modules will work with data structures tuned to the schema. Let's take a fragment from a book DTD as an example:

```
<!ELEMENT book (title, author, isbn)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
```

A schema compiler that works with Java can automatically define a class that represents books. A simplified version of how such a class might look is shown in Listing 2. In actual cases the class definition will probably include getter and setter methods and some other niceties that can be safely ignored for the purposes of this example.

Consider an example XML fragment:

```
<book>
  <title>XML: A Primer</title>
  <author>Simon St. Laurent</author>
  <isbn>076453310X</isbn>
</book>
```

When the deserializer module generated by the schema compiler parses the XML, it will create an instance of a Book object that will be equivalent to the one created by the following Java fragment:

```
Book aBook = new Book();
aBook.title = "XML: A Primer";
aBook.author = "Simon St. Laurent";
aBook.isbn = "076453310X";
```

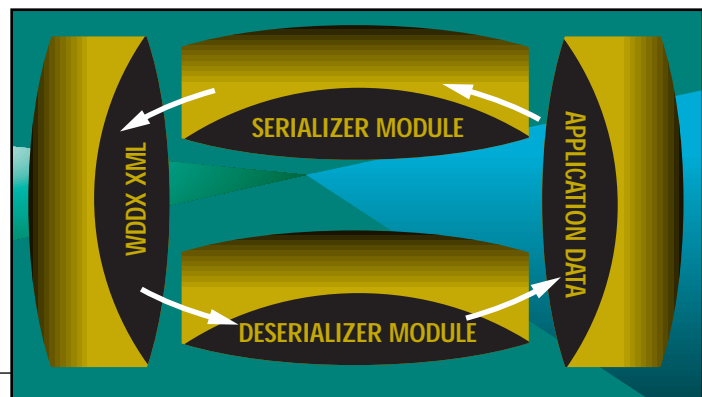


FIGURE 1 WDDX serialization/deserialization process. The serializer/deserializer modules are provided by the WDDX SDK.

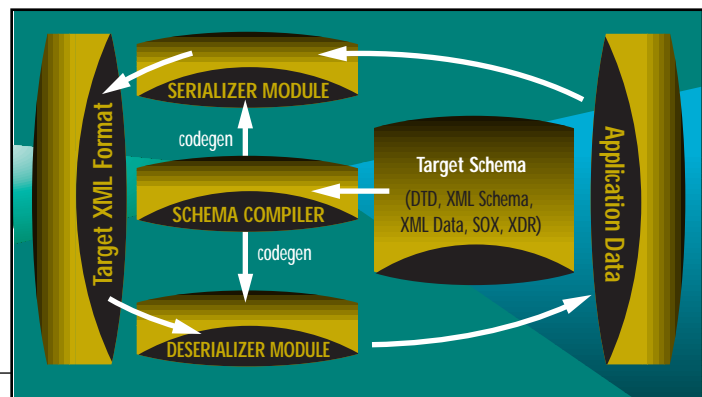


FIGURE 2 Serialization/deserialization process with a schema compiler. New serializer/deserializer modules are generated for every new schema.

Software AG

www.softwareag.com/tamino

Because DTDs cannot specify many interesting data types, not even numbers, schema compilers won't become widely used until the W3C XML Schema activity produces a final specification, part of which will specifically cover datatypes. Therefore, don't count on robust schema compilation technology for at least 6–9 months. One of the notable efforts in this space is the XML Data Binding activity that is part of Sun's Java standardization effort. The folks at Sun, together with a bunch of XML experts, including yours truly, are working on a schema compiler for Java. No public information has been released yet so I have to keep relatively quiet on the subject.

Schema Translation

Schema translation refers to the conversion of XML documents from one format to another. It is also known as XML integration/conversion. Schema translation is very important in the context of B2B because the world of business is highly heterogeneous. No single organization owns or will ever own the standards that specify how the information relevant to B2B processes is going to be represented in XML. In fact, right now there are at least two broad efforts to manage all kinds of B2B XML standards. There are also hundreds of efforts focused on vertical business niches.

BIZTALK

The first of these broad efforts is BizTalk. This is a Microsoft initiative to drive the adoption of XML for e-commerce and application integration. The BizTalk Framework establishes guidelines for the successful use of XML for B2B. BizTalk.org is a repository of schemas related to different e-commerce segments that conform to the BizTalk Framework. Biztalk.org is also a community of companies using XML for B2B. Microsoft is working on tools that will facilitate the use of BizTalk schemas and the integration of business processes based on BizTalk.

OASIS

The Organization for the Advancement of Structured Information Standards (OASIS) is an international consortium focused on managing open standards for content and data interchange. Probably the most relevant initiative at OASIS is ebXML, the Electronic Business XML initiative, which is an effort to establish a global framework for the exchange of business data. The OASIS repository for XML schema is xml.org. One can find a lot of interesting specifications there. For example, see IBM's submission of the Trading Partner Agreement Markup Language (tpaML).

By now you must realize that there will be many specifications produced by different standards bodies covering similar areas of e-commerce. Since interoperability is crucial to B2B, schema translation tools must be employed to facilitate B2B processes that work with multiple standards.

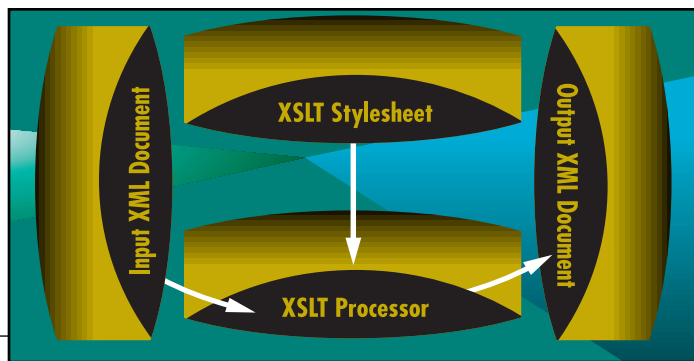


FIGURE 3 Operation of an XSLT processor

There are two main approaches to schema translation: one utilizes custom software and the other uses XSL transformation. The custom software approach relies on you parsing the incoming XML, building some data structures that represent the data specified in the XML and then generating XML in the prescribed format.

XSL TRANSFORMATIONS

The second approach centers around the Extensible Stylesheet Language Transformations (XSLT) specification.

XSLT is the standard mechanism for transforming XML from one format to another. An XSLT processor takes an input XML document and, using rules from a stylesheet (another XML document that specifies how the conversion is to be performed), generates an output XML document. There are several freeware XSLT processors. You can obtain XT from James Clark's site (www.jclark.com) – James is the editor of the XSLT specification. Xalan is an implementation that was started by IBM's Alphaworks division. It has since been contributed to the open source initiative at xml.apache.org.

Writing custom schema translation code or XSLT stylesheets for schema translation can take a lot of effort. To ease the pain, vendors are doing one of three things. Some are adding out-of-the-box translation tools for popular schemas. Others are defining metalanguages that specify at a higher level how schema translation should be done. Developers write an XML document that specifies how elements, attributes and content of the source document are mapped to elements, attributes and content in the target document. Some type of a code generator will then build the custom code or XSLT stylesheet that does the actual work (see Figure 3). This approach is similar to that of schema compilers. The difference is that schema compilers work with XML and application data while the schema translation tools

LISTING 1

```
<wddxPacket versi on="1.0">
  <header/>
  <data>
    <recordset rowCount="2"
      fi el dNames="url , headl ine_text, source">
      <fi el d name="url ">

<string>http://d.moreover.com/cl ick/here.pl ?x5835287</string>

<string>http://d.moreover.com/cl ick/here.pl ?x5834977</string>
      </fi el d>
      <fi el d name="headl ine_text">
        <string>Cuban diplomat, expelled from US, must
        leave Canada</string>
        <string>Sharif Lawyers to Boycott Trial</string>
      </fi el d>
      <fi el d name="source">
        <string>CNN</string>
        <string>New York Times</string>
      </fi el d>
    </recordset>
  </data>
</wddxPacket>
```

LISTING 2

```
public class Book
{
  // These are the properties of books
  public String title;
  public String author;
  public String isbn;

  // This is how we generate XML from a book object
  public void toXML(xml OutputStream xos) { /* details */ }

  // This is how we construct a book object from XML
  // Think of this as a factory method
  public static Book fromXML(xml InputStream xis) { /* details
  */ }
}
```



work exclusively with XML. Finally, some vendors are adding GUI tools for schema translation. As a developer, you drag and drop elements and attributes between the source and target documents to specify how you want the translation to be done. Code generation then happens behind the scenes just as in the previous approach. Most application server vendors (Allaire, BlueStone, IBM, Microsoft, SilverStream) have announced initiatives in this area. The details are scant and quickly changing.

Tried and True

To finish, here are some simple guidelines for developers that can help steer you in the right direction when you want to apply XML to B2B e-commerce:

- Analyze the problem space from three different views: business process, syndication and function.
- Choose a reliable mechanism for delivering XML. Don't forget to worry about security and transactions. (Unfortunately, a solid discussion of reliable XML transport is outside the scope of this article.)
- If the focus is on data, choose technologies that facilitate automatic XML data mapping such as WDDX.
- If the focus is on a particular XML format, choose schema compilation tools that can generate serialization/deserialization modules for this format.
- If the focus is on integrating different XML formats, choose XSLT processors or other schema translation tools.
- If the focus is on exposing remote APIs for application integration, choose data-centric technologies that can handle request/response processing, e.g., WDDX, XML-RPC, LDO and SOAP.
- Minimize custom coding, particularly for parsing XML and generating XML from application-level data structures.
- Keep a simple approach and don't rely much on draft specification.

Last, always bear in mind that the B2B space is growing at an amazing rate. Try to keep up with it as best you can. ☒

XML References

- **BizTalk:** BizTalk. Microsoft Corporation. See www.biztalk.org.
- **ICE:** Information Content Exchange (ICE). W3C (World Wide Web Consortium). See www.w3.org/TR/NOTE-ice.
- **LDO:** Lightweight Distributed Objects (LDO). Casbah.org. See www.casbah.org/LDO.
- **SOAP:** Simple Object Access Protocol (SOAP). Microsoft Corporation. See <http://msdn.microsoft.com/xml/general/soapspec-v1.asp>.
- **SOX:** Schema for Object-Oriented XML 2.0. W3C (World Wide Web Consortium). See www.w3.org/TR/NOTE-SOX.
- **WDDX:** Web Distributed Data Exchange (WDDX). Wddx.org. See www.wddx.org.
- **XML Data Binding:** JSR-000031 XML Data Binding Specification. Sun Microsystems. See http://java.sun.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html.
- **XML-RPC:** XML-RPC. xmlrpc.com. See www.xmlrpc.com.
- **XMLSchema: Datatypes:** XML Schema Part 2: Datatypes. W3C (World Wide Web Consortium). See www.w3.org/TR/xmlschema-2/.
- **XMLSchema: Structures:** XML Schema Part 1: Structures. W3C (World Wide Web Consortium). See www.w3.org/TR/xmlschema-1/.
- **XSLT:** XSL Transformations (XSLT). W3C (World Wide Web Consortium). See www.w3.org/TR/xslt.

AUTHOR BIO

Simeon (Sim) Simeonov, chief architect, Allaire Corporation, has been developing software for more than 15 years. His areas of expertise encompass object-oriented technology, compiler theory, Internet tools, enterprise computing and the broad spectrum of XML technologies. In his current role at Allaire, Sim provides direction for the evolution of the company's technology and product strategy as well as the architecture of Allaire's e-business platform products: ColdFusion, HomeSite, JRun and Spectra. Sim is currently working on architectures for the next generation of distributed Web applications.

SIMEON@ALLAIRE.COM

ComputerWork.com

www.computerwork.com



*A progress report on the quest to help with
moving EDI to an XML/Internet base*

Introducing ebXML

OASIS is involved in quite a range of XML activities. In my last column I discussed its increasing role in helping to standardize XML applications and talked about what's being done with XML.org. In this column I'll introduce you to the Electronic Business XML Initiative (ebXML), OASIS's joint initiative with the United Nations/CEFACT group.

Let's begin with some background related to e-commerce. First, it's not a new idea – companies have been buying and selling goods electronically for 20 or 30 years. They weren't using the Internet or XML back then, of course, but rather EDI – electronic data interchange. This technology, usually associated with rigid, predefined message sets and value-added networks (VANs), is still in wide use today.

The VANs are private networks that guarantee message delivery and provide a certain level of security. For cross-industry commerce the two major dialects of EDI messages are ASC X12 and EDIFACT. X12 began life as an essentially North American standard while EDIFACT covered much, though not all, of the rest of the world. The group, now called UN/CEFACT, developed and continues to maintain EDIFACT. Since cross-border mergers and acquisitions are not uncommon business activities, some companies use both message format specifications for dealing with trading partners.

EDI is big business – especially for big businesses. It's been estimated that 98% of the Fortune 1,000 use some

form of EDI and that the overall use of EDI is growing. The cost savings in terms of time, money and effectiveness are significant, and it's a reliable way to transmit electronic messages up and down your supply chain. EDI is also a good business for companies that provide EDI programming, support and VAN services. (Here's my regular disclaimer: I work for IBM, one of the leading companies providing EDI services; the opinions expressed in this column are my own.) So what could possibly be wrong, given the apparent widespread acceptance of EDI and the evident business advantages for everyone involved?

While EDI use – according to Gartner – is indeed growing overall, the *rate* of growth is slowing. Adoption remains strong in the big companies, but outside the Fortune 1,000 the implementation rate is estimated to be as low as 5%. That's a big difference, especially when you think about the players in a typical supply chain. At some point in the chain the programming, maintenance and VAN costs get too large for small- and medium-sized enterprises (SMEs). So what we see are electronic messages moving back and forth among the big

players, and faxes, phone calls and paper used among the smaller ones. I'll leave it to you to think about the reliability aspects of receiving a faxed purchase order and then typing the relevant portions into your in-house order-tracking system. To make matters worse, that purchase order was probably printed from a computer before it was faxed. All this clearly adds cost and time to the process of buying and selling goods.

Ray Walker, chair of the UN/CEFACT group, adds another important perspective when he reminds us how easily developing countries, as well as SMEs, can be left out of the electronic B2B (business-to-business) world. If the cost of the infrastructure is too high, the number of potential partners with whom you might do business electronically is limited accordingly.

So it's not altogether surprising that people have turned to the ubiquitous Internet and its related technologies to help reduce the expense of connecting to as many trading partners as possible. One of those related technologies is XML.

Helping Move EDI Forward

Traditional EDI messages are broken into sections, subsections and fields, with separators between them. The message format is rigid in the sense that information must be placed

in certain locations or it is ignored or rejected. The data isn't self-describing, but it isn't particularly verbose either, except that parts of general-purpose messages may be left empty in specific scenarios. The skills and tools related to working with EDI messages are specialized and therefore expensive. This is a great place to apply XML for message formats. Companies like XML Solutions (www.xmlsolutions.com) are doing just this, and organizations such as the loosely knit XML/EDI group have studied how XML can best be used in moving EDI forward and making it more pervasive.

The value of XML hasn't been lost on EDI users in vertical industries. Health Level Seven (www.hl7.org) has been working on an XML representation for version 3 of its specifications. HL7 and OASIS recently exchanged reciprocal memberships so that the XML technical work of each can be more widely and effectively shared throughout the industry.

Note that moving EDI to an XML/Internet base is not the only XML activity in e-commerce today. Companies such as Ariba and Commerce One have used XML and the Internet to create procurement solutions. While auctions, marketplaces and exchanges are all commerce models that are rapidly becoming important and profitable, EDI wasn't designed to support these kinds of transactions.

If you look at all the cross-industry and vertical industry messaging solutions, there's a tremendous amount of redundancy and repetition of standards development around infrastructure. Take the message envelope that surrounds the actual domain-specific payload of a business message (for example, a purchase order, a stock transaction, a lab test request or a plane reservation confirmation). This envelope probably contains information concerning who originated and who is to receive the message. Beyond this, the information in the message envelope can vary quite a bit in content and format.

Microsoft's BizTalk, Open Buying on the Internet (OBI), RosettaNet and the EDI message specifications all use different message envelope representations. B2B software that operates across enterprises must have components to read and parse each of these formats, thereby adding to the base cost of the applications. When still another message envelope format is developed, maintenance and upgrade costs will increase accordingly.

Defining Universal Specifications

It's unrealistic to think that a single vendor or a narrowly scoped industry standards consortium can define a universal message envelope format that then gets adopted across multiple industries. But perhaps a broad-based consortium of companies, trade organizations and industry groups has the wherewithal to define specifications such as this and to actually get them widely implemented.

It's the hope of OASIS and the United Nations/CEFACT group that ebXML is just such an initiative. The ebXML effort was announced in September 1999. Klaus-Dieter Naujok of Harbinger Corporation and the UN/CEFACT group is the chair and I represent OASIS as the vice chair. The mission of the initiative is "to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties." The official Web site is at www.ebxml.org.

Our first meeting was held in San Jose, California, in November and was attended by more than 120 people rep-

Each project team has its own page on the ebXML Web site describing its mission, goals and progress. The technical architecture team encompasses the "big picture" of our technical work. Here's how it describes core components, for example:

"Core components are elements of the component library that are common to multiple business domains. These core components are used within domain-specific components and are also specialized for domain-specific requirements."

Although these teams had already done some work in the two months following the first meeting, things got going for real at the second meeting in Orlando at the end of January 2000, which took the form of three days of team meetings surrounded by two half-day plenaries. The team meetings were exhaustive analyses of prior art in each area and strategy sessions for how to move ahead. Let me emphasize that the ebXML effort isn't trying to create all the technology it needs from scratch but rather to borrow the best existing tech-

If you look at all the cross-industry and vertical industry messaging solutions, there's a tremendous amount of redundancy and repetition of standards development around infrastructure

resenting more than 50 organizations from around the world. It was largely a "show and tell" meeting during which the EDI people and XML e-commerce people explained to each other what they'd been doing and what they hoped ebXML might accomplish.


As you might expect, the meeting wasn't without its share of politics. But I'm happy to report that the group came together to create eight project teams whose purpose is to define and implement the work of the initiative over the planned 12- to 18-month time frame. The project teams are shown in Table 1.

- ebXML requirements
- Business process methodology
- Technical architecture
- Core components
- Transport/routing and packaging
- Registry and repository
- Technical coordination and support
- Marketing, awareness and education

TABLE 1 ebXML project teams

nology, which will then be combined with new work to create a cohesive infrastructure for e-business.

In order to be relevant to the development of e-business, the ebXML effort needs to produce its specifications in a timely manner and must make drafts available every few months. The first draft specifications will be available after the Brussels meeting in May. We're actively soliciting the help of organizations such as the IETF to assist in the rapid development of the ebXML framework. Participation is open to everyone and you can register yourself for project team participation at the Web site.

In my next column I'll discuss a complementary technology that's being standardized within OASIS/XML.org called Trading Partner Agreement Markup Language. You can start learning about it by reading the initial submission from IBM at www.xml.org. 

AUTHOR BIO

Bob Sutor is IBM's program director for XML technology and a member of the OASIS board of directors. While a member of IBM's research staff, Bob developed applications and led advanced technology projects related to XML and Internet publishing.

SUTOR@SUTOR.COM

THE SPELL OF XML: THE NEXT INTERNET REVOLUTION

It's changing the
way people build
Web sites –
and what they
can do with
them



One of the most important developments for the Internet, XML is enabling businesses to take their corporate information to the Web with an efficiency and richness that heretofore hasn't been possible. Most important, it is rapidly becoming the lingua franca for conducting business on the Web. It is the format for conveying product information, the means by which transactions are conducted and the glue that enables enterprise applications to talk to each other.

You should find XML of interest if you create Web sites for a living or are involved in maintaining your company's presence on the Internet, because you'll want to learn how XML can be used to provide more functional Web sites than HTML alone can. If you're responsible for managing corporate documents, you'll want to learn how XML can be used to create a robust, scalable repository with possibilities for entirely new means of delivery. Or perhaps you're just interested in HTML and the Internet, and want to know how XML is going to change the future of the Web.

This article explains what XML is and what it's good for. Numerous examples are presented to show that XML solves real-world problems and is actually not all that difficult to understand. It describes the status of XML today, where it's going in the future and what you should be doing today to get ready for it. As the CTO of SoftQuad Software, I'll also take the liberty of presenting SoftQuad's role in the development of XML, along with the things we're doing to bring the benefits that XML can achieve to a broad range of users.

Less emphasis is placed on the strictly technical aspects of XML as these are well covered in a number of references. (A good starting point for delving into those details can be found, for example, on SoftQuad's XML resource page – see "XML Reference" at end of article.)

What Is XML?

XML is most easily understood by comparing it to HTML. HTML files consist of text that has markers (tags) to describe, for example, how the text should be presented. In the following HTML text:

The **real** reason for doing this...

the **** and **** tags around the word *real* indicate that it should appear in bold face. Other HTML tags indicate the meaning of a piece of text. For example:

<TITLE>SoftQuad Home Page</TITLE>

indicates that the title of the document is SoftQuad Home Page. A Web browser will recognize this and use it for the title of the browser window. Other HTML tags are used to put images on a page and to create links to other documents.

HTML is a fixed markup language – you have a certain set of tags available and no more. The purpose of XML is to provide a way to define new sets of tags, hence the name *eXtensible* Markup Language. For example, you might want to use XML to define tags that describe a customer record:

<Customer Name="Bruce Sharpe" MemberID="1AB345">

The tags in HTML serve a number of different purposes, but most have to do with formatting and presentation. HTML has been spectacularly successful at this because it's quite simple but still rich enough to create good-looking pages. What HTML is not very good at is indicating what a piece of information is. You won't find a **<symptom>** tag for doctors, or a **<part-number>** tag for mechanics, for example. XML fills this gap. It gives you the ability to define tags that are meaningful for particular purposes and in particular domains, whether finance, engineering, linguistics or chemistry. It allows you to denote not only what the various parts of a document are, but also what various bits of data are. This simple feature has profound implications for data-driven Web applications and information delivery, as we'll see below.

What Is XML Good For?

XML can be used in a number of ways. The following sections provide the flavor of the most important categories, with examples in each case for concreteness.

1. XML PROVIDES A ROBUST FORMAT FOR CONTROLLING APPLICATIONS.

Some of the earliest real-world examples of XML are those in which it's used to define an entirely new format for a special-purpose application. Microsoft's Channel Definition Format (CDF) is an XML-based format that's used to describe active channel content. Web authors create CDF files that Internet Explorer reads to set up a user's channel for that author's site.

An example of a portion of a CDF file is shown in Listing 1. It describes where the channel pages are, what logos to use in the Channel Bar, and so on.

For such special-purpose formats it's expected that special-purpose applications, such as the Channel Crafter application shown in Figure 1, will become available to make the process of creating and managing them as easy as possible.

2. ROBUST XML MARKUP ENABLES AUTOMATED PROCESSING.

Because XML formats are precisely defined and predictable, they're a natural choice for situations in which you want to exchange data that is to be processed automatically. For example, many people have contact information in their e-mail signature, and rely on this to make sure friends and colleagues have the latest information. Wouldn't it be handy if there were a way to get the information out of those signatures and automatically update your contact database? Suppose people agreed on an XML format for signatures that looked like the following code:



FIGURE 1 Standardized formats allow for specialized interfaces

```
<signature>
<name>Jane Doe</name>
<voice>555-123-4567</voice>
<fax>555-123-4568</fax>
<email>jane@nowhere.com</email>
</signature>
```

It's not hard to imagine a program that could comb through your e-mail files, pull out this signature information and put it into your contact database. There are several advantages to using XML for this application. The relevant information is unambiguously marked up with XML tags, so that you wouldn't get some other set of information put into your contact database inadvertently. For automatic processing of a set of files, it's important to make the data format as robust as possible so the data remains valid even if some details of the format change in the future. A comma-separated format such as:

Jane Doe, jane@nowhere.com, 555-123-4567, 555-123-4568

wouldn't have the same level of robustness since the order of the information determines the relevance and meaning. An XML format, on the other hand, is self-describing so new fields can be added without destroying the validity of the existing fields.

Figure 2 shows what it might be like to edit such information with a general-purpose XML editor. This is a simple example of exchange of data for automated processing. But the use of XML is growing rapidly for B2B transactions, e-commerce and other kinds of data exchange. In the past, specialized binary formats were invented for applications like this. An important advantage of XML is that it is normal text, unlike binary formats that are often proprietary and not human-readable. Text-based programs like e-mail have no problems with XML.

3. XML CAN EXTEND HTML.

XML enables you to add tags to HTML for customized processing by, for example, a server-based application. SoftQuad's own HoTMetal Application Server is an example of a program that can process XML by adding tags from Miva, an XML-based language, to your HTML documents. These tags enable scripting, database access and commerce to be added to your Web pages (see Figure 3).

4. XML ENABLES THREE-TIER WEB APPLICATIONS.

One of the most important uses for XML is expected to be the integration of data from sources that are dispersed and exist in a variety of formats. Because XML markup maintains the intelligence of the data all the way through a processing chain, data can be retrieved from several sources, combined and customized, and sent to another level of processing. A demonstration of such an application can be found in the frequent flyer pages of the fictitious Softland Airlines. This demo makes use of the XML capabilities of the HoTMetal Application Server (HMAPPS) and Internet Explorer 4.0 or later.

The scenario is that a member of Softland's frequent flyer program visits their site to see if he or she has enough points for a flight to Paris. The member is presented with information about Softland's standard awards and about award specials available from Softland's partner airlines. Information about the member is used to provide a customized view of the data from Softland and its partners, and an interactive exploration of possibilities.

Figure 4 shows the architecture of the site. When the member signs in, HMAPPS retrieves his or her information record via ODBC from the Softland member database. It's then converted and sent to the browser as an "XML data island" embedded in the HTML data stream. The member information XML looks like this:

```
<XML ID = "customerInfoXML" TYPE = "text/xml">
<CUSTOMER
ID = "TheMember"
```

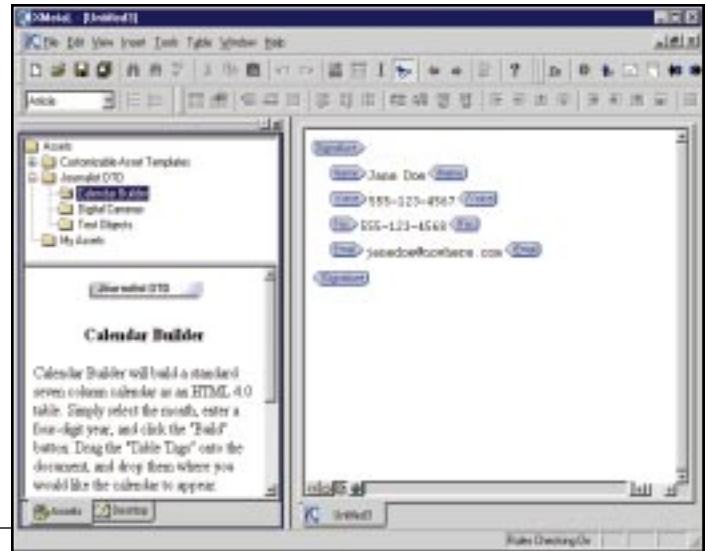


FIGURE 2 Editing with a general-purpose XML editor

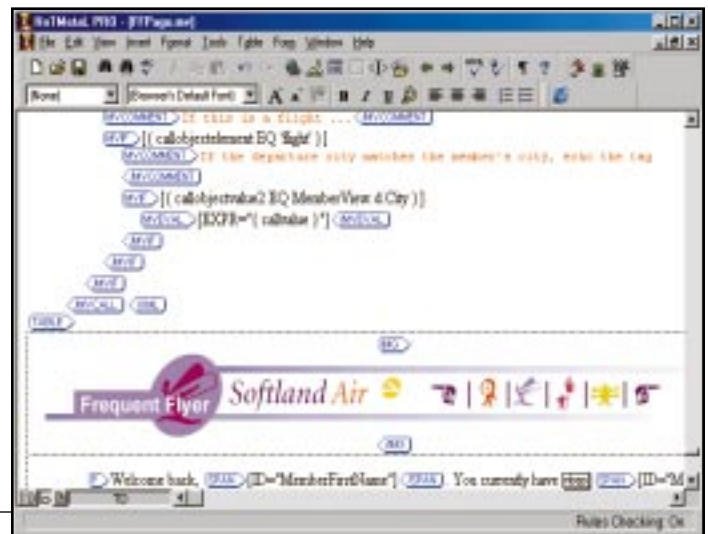


FIGURE 3 HoTMetal PRO can be used to edit HTML (tags starting with MV).

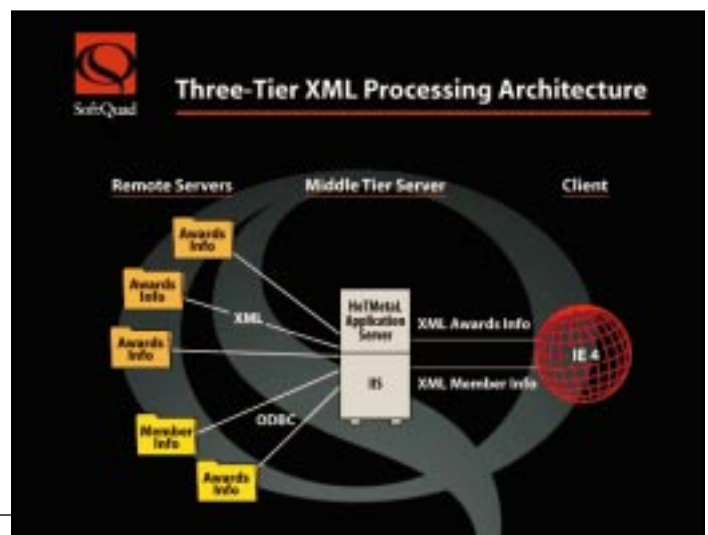


FIGURE 4 XML data provides an interactive experience in the browser.

VSI

www.vsi.com/breeze

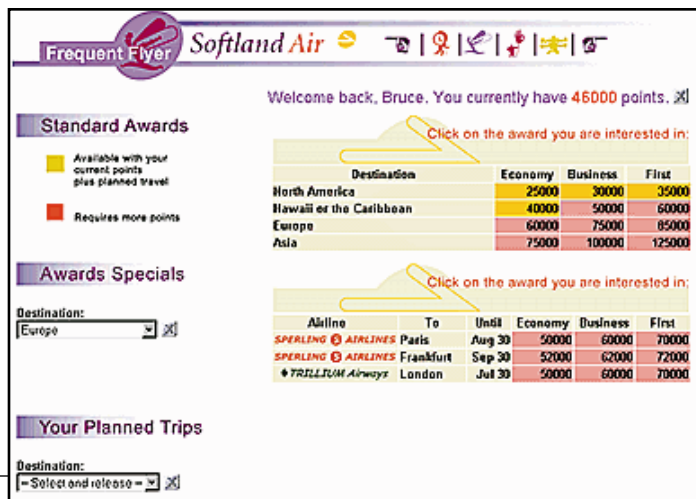


FIGURE 5 XML enables data integration on the middle tier.

```
MEMBERID = "1AB345"
FIRST = "Bruce"
LAST = "Sharpe"
POINTS = "46000"
CITY = "Vancouver"
CONTINENT = "NA" />
</XML>
```

Award specials information comes from two types of data sources. The first is the databases of the partner airlines, Sperling and Trillium. They are searched for flights that go to Europe and originate in the member's usual departure city as found in the member profile. The data is delivered in XML (XML data comes from the back-end resource/database) and combined into a single XML data island that's sent to the browser in the HTML stream. Dynamic HTML is used in the browser to select records that have a destination of "Europe" and format them into a table. The second data source is for bonus awards – flights that provide more than the usual number of points to common destinations. All the flights that originate in the member's city are found for both partner airlines and combined into a single XML data island that's sent to the browser. They are then formatted into a table via DHTML (see Figure 5). Users can then experiment with "what if?" scenarios to see what their points situation will be if they take certain flights. When a flight is selected, the member's current points total is updated to reflect the points that would be accrued from that flight and the awards tables are appropriately color-coded.

Using this architecture, an airline builds a frequent flyer Web site that's fully interactive and provides a productive and pleasurable experience for customers visiting it. The personalized data clearly conveys how many points a member has and exactly what trips he or she qualifies for. A member can also determine how many points are required for any given trip and how best to acquire them. Through this interaction customers are encouraged to increase their "paid for" travel and to stay loyal to their airline.

In this example XML is used as a structured information interchange protocol. It enables information from multiple remote databases to be aggregated and personalized, then delivered to client browsers based on end-user requirements. The original information is stored in databases in whatever format the database uses. Without XML, data retrieval particular to each database would have to be implemented. The problem here is that you can't easily change what information you want or how it should be combined. It's much easier to combine data that's tagged with information saying what it is, so XML makes this process much more robust and flexible. XML enables data integration because it provides a common format for exchanging data from heterogeneous sources.

5. XML IS THE NATURAL CHOICE FOR INTEROPERABLE FORMATS.

- **SMIL:** XML is rapidly becoming the language of choice for defining interoperable formats. Some formats, such as the Synchronized Multimedia Integration Language (SMIL), are designed to store content the way HTML does. SMIL was designed by a group within W3C to allow integration of independent multimedia objects into a synchronized presentation. It includes tags describing the layout of the objects and hyperlinking between objects, as well as the behavior of the presentation with time. The sample SMIL file in Listing 2 is taken from the W3C specification. This example overlays the image in the region with the ID value of c1 with the image with the ID value of c2.
- **RDF:** Other formats based on XML, such as the Resource Description Framework, store other types of information. RDF is designed as a general way of storing metadata – information about information. It's used as a base for several formats, such as the Dublin Core set of information used to describe Web pages to enable finding the document that best matches given criteria. For example, you might want to find books written *about* Jane Austen rather than *by* Jane Austen. If you look for "Jane Austen" in a standard search engine on the Web, you'll find many books *by* Jane Austen as well as the books *about* Jane Austen. Being able to specify what you want more precisely will help you find it.

The list of potential criteria to search by includes allowing qualification of the descriptions. An example would be at an online book club where reviews are written about books. The Dublin Core schema for RDF could be used to give information about the author of the review as well as the review itself. Since the Dublin Core is a standard description of information, search engines could gather the information from any site and make it available. Web authoring tools can provide easy ways to fill in the information for Web documents, thus making it easier for people using search engines to find the right pages.

Although the Dublin Core schema hasn't been finalized yet, the following example shows how a simple description might look:

```
<RDF:Description
RDF:Href="http://www.bar.com/some.doc">
  <DC:Creator>John Smith</DC:Creator>
</RDF:Description>
```

This contains the location of the document (called a *resource* in RDF) and the creator (the author) of the document. Other useful information is added in a similar way. RDF provides a schema for Dublin Core, that is, it defines which tags are used in Dublin Core. The "RDF:Description" tag tells an application, such as a search engine, that this is an RDF tag "Description" with a special meaning, defined by RDF. The "DC:Creator" tag tells an application that the Dublin Core schema has a tag "Creator" with a special meaning. If an application understands what "RDF:Description" and "DC:Creator" mean, it may do some special processing of the tags. If it doesn't, it'll usually show the contents of the tags to the reader of the document.

6. XML DOCUMENTS MAKE MANAGEMENT OF UPDATES EASIER.

The examples we've given of XML so far have shown its use in small data chunks or as a few tags added to HTML. However, XML is also quite capable of being used for full-blown documents that aren't based on HTML. For certain types of documents the structure and order that XML markup language brings can enable much easier management of versions and updates.

VCR USER MANUALS

For example, suppose you work for a manufacturer of VCRs and are responsible for managing end-user documentation. The documentation is supplied as a printed manual, but you also want to make it available in the customer support area of your Web site. In particular, a typical customer has trouble figuring out how to set the clock, and the blinking "12:00" threatens to completely wreck domestic peace.

The challenge for you as documentation manager arises because your user manuals are about 80% the same for all models, but the instructions for setting the clock are almost completely different from model to model: some use onscreen programming, some require pushing buttons on the front panel, and so on. Meanwhile, customer feedback and an internal quality improvement campaign are producing a steady stream of requests to update the manuals for one reason or another. Managing all this cries out for a database solution of some kind, but a traditional relational database is an awkward fit to a documentation task.

XML is tailor-made for such a situation. (More precisely, XML derives from a heritage of documentation solutions.) If the manuals are written in XML, appropriate tags can be created to distinguish information that's specific to particular models. Information that's common across models can be shared. Listing 3 is a sample of what part of the manual might look like in XML.

The information about how to set the clock on your VCR is kept in a single master file, together with the information about how to set clocks on any other VCR the manufacturer produces. This means there's only one document to worry about for this function. Every time a new model is added to the line, the instructions for setting the clock are added to the master document.

Some steps in the process are the same as for other models, so nothing new needs to be written there. Some steps are only for newer models, some only for a particular model. XML lets you mark the information according to whether it's applicable to a model range (the "modelfrom" and "modelto" attributes on process), specific models (the <var model="CA">) or all models that don't have specific instructions (<var model="other">). Any combination of these is also possible.

It's then easy to write a script that shows the customer only the parts of the owner's manual that apply to the model number he or she has as well as those that are applicable to all VCRs. It's also possible, based on the information given, to change the formatting of the document. For example, images of the correct buttons can be shown, based on the content and attributes on the <button> tags. The online versions of the manual can be created on the fly, customized precisely to the user's needs. This type of dynamic content assembly is becoming very popular on the Web.

SOFTWARE DOCUMENTATION

My company, SoftQuad, writes all its product documentation for end users in XML. This has a number of advantages. We can write a master document and indicate the information that's relevant to only one platform. We can make sure the documentation contains all the necessary parts because we can set up our XML authoring system to validate the documents. This validation process checks the document against a formal grammar definition of the tags and their relationship to each other, such as which elements can contain which other elements. When we know the document is valid and everything is in the right place, we can pass the documentation on for further processing.

We generate printed documentation and Windows help files from the same XML source. Since we know the document obeys the rules we set up for it, we don't need to worry so much about error conditions in the conversion process. We know that the abstract will be in the right place, for example, and that each chapter will have a title. The process runs smoothly no matter which product is being documented. The documentation specialists concentrate on the content of their writing, and the layout and design specialists concentrate on making sure that the document looks good on paper and on the screen.

The benefit of using XML here is that we need to write the documentation only once for each product, and we can reuse and republish it many times in different formats.

XML Today

XML is derived from SGML (Standard Generalized Markup Language), an ISO standard developed in 1986. During the last 10 years the SGML community discovered that many of the features of SGML, though useful for some applications, weren't always necessary. These were the features that were taken out of SGML to create XML, making it easier to create

applications that process an XML document. Since XML is a pure subset of SGML, people who discover that they need the more advanced features available in SGML will be able to convert their systems to SGML and won't lose the investment they have made in their documents and data.

As with SGML, you can provide a formal definition of an XML tag set via a document type definition (DTD). An alternative mechanism called *schemas* has been proposed as well.

XML is a W3C standard. This means it's stable and expected to be widely deployed. It also means that it can be supported by several vendors, secure in the knowledge that XML 1.0 will be the same for quite some time. We can look forward to a host of tools that can be used in combinations that no single vendor might have thought of. Moreover, XML will scale, so you preserve the investment in your data even when you upgrade your tools.

Many software companies are now working on XML products. Already there are a number of XML parsers available, in Java and C++. Microsoft first provided support for XML in Internet Explorer 4.0, extended it considerably in version 5.0 and continues to provide updates to keep pace with the evolution of the standards. There are also a large number of XML tools available on IBM's and Oracle's developer sites.

Future XML

XML alone is the basis for the examples discussed above. However, several other standards are currently being defined that will complement the language itself and broaden its application.

STYLES

If you're using XML only for data interchange, you may not care about formatting it for display. But sometimes you do want to display it. Since any element in XML can have any name, unlike HTML, there needs to be a way to describe what the document should look like in a browser or editor. The display of XML is done through stylesheets, which specify to an application which elements should be in a bold typeface, which should start on a new line, which should be hidden, and so on. There are two stylesheet languages for the Web.

CSS (Cascading Style Sheets), originally developed by W3C for use with HTML, has been extended for use with XML. CSS can be used to override the default display of HTML to allow extensive control over the appearance of a page. A fairly simple language without any scripting capabilities, it does most of what people need without being overly complicated to use. The latest version of CSS, CSS2, is a W3C Recommendation. CSS3 is currently under development by the W3C. The W3C site lists a number of authoring and browsing tools that support applying CSS to XML.

XSL (eXtensible Stylesheet Language) is a more sophisticated and powerful style language for XML. When people talk about XSL today, they're usually referring to XSLT (XSL Transformations), a subset of XSL that provides a way of transforming XML to another format. An increasingly popular way to deliver XML content to a browser is to first use XSLT to transform it to HTML, then send the HTML to the browser. XSLT is a W3C Recommendation, but the full XSL stylesheet language is still under development.

LINKS, QUERIES AND MORE

There are a large number of specifications related to XML at various stages of development in the W3C. In some cases there are implementations of these in commercial products that are based on working drafts of the specifications.

XLink provides a way to insert tags in XML documents that create and describe links. One of the most important aspects of the Web is the ability to link documents together. XLink is designed not only to allow the simple linking capabilities of HTML to be applied to XML documents, but to enhance them.

XQL is a proposal for regarding XML documents as a data repository and provides a language to express queries on them. It's been implemented in some commercially available products and is being considered by the XML Query working group of the W3C.

Other specifications in the works will provide mechanisms to address parts of an XML document and ways of handling fragments of documents.

DOCUMENT OBJECT MODEL

DOM is one of the most important specifications being worked on in the W3C. It provides the interface that enables a program written in Java, JavaScript or some other language to access and manipulate a Web document, be it HTML or XML. This will standardize dynamic HTML, and extend the functionality to XML as well. DOM Level 1 is a W3C Recommendation that has been implemented by such products as Internet Explorer and SoftQuad's XMetaL. DOM Level 2 is nearing completion.

EXTENDING HTML

HTML became a standard for the Web because every HTML editor could create a document that every HTML browser could display without the user having to do anything. This isn't necessarily the case anymore for XML, since authors can define the tags any way they want to. Sometimes you don't need to create everything from scratch; often all you need is your new elements added to HTML. The XML block in the airline demo above shows one potential way for embedding an XML data island into HTML. W3C is working on general solutions for embedding XML in HTML. Also, HTML itself has been reformulated in an XML-compliant way in XHTML. This promises to bring the rigor of XML to HTML while maintaining compatibility with existing HTML browsers.

SCHEMAS

One of the most keenly anticipated W3C specifications is XML Schemas. Today, in order to describe the tags that are available for a particular XML document, the only mechanism available is the DTD. While adequate for many documentation purposes, a DTD has limitations when XML is being used for a more general data interchange. One important example of such interchange is e-commerce. Schemas are intended to address these limitations. For example, often it's not enough to create a tag to describe a date – you'd also like to specify what date format is allowed. Or in the case of a tag expressing a quantity, such as an inventory level, you might want to specify what range of numbers is valid. Schemas will let you do all this and more.

Several key mechanisms for using XML in B2B commerce, such as Microsoft's BizTalk and Commerce One's xCBL, are based on schemas

that have been defined in advance of the final specification. The vendors have committed to revising their use of schemas to conform to the W3C recommendation once it becomes available.

Industry Definitions

Industry-specific document definitions are going to be very important. Just as the definition of HTML allowed for specialized tools to become available for creating HTML documents, so will the definition of industry-specific document formats allow the creation of specialized tools for those documents and specialized macro sets for more general tools. Numerous industry groups are collaborating to create public DTDs for particular purposes, ranging from commerce to astronomy to general content aggregation.

SoftQuad and XML

As far as my own company, SoftQuad, is concerned, we've been involved in the development and application of XML since its inception, having participated in the development of SGML (the parent of XML) and DSSSL (an ancestor of XSL). We developed one of the first SGML authoring tools, Author/Editor, and are now transferring that knowledge and experience to XML.

As a member of W3C since that organization's inception, SoftQuad has taken part in many of the XML-related working groups in W3C – XML, XSL, CSS&FP, HTML and RDF – and also chairs the DOM working group. There's a lot of work going on within the W3C and other consortia to define XML standards, document types and schemas. Whether you're looking for a standard way to describe purchase orders and other general business documents, or content for specific industries such as insurance, real estate or finance, there's sure to be an industry group working on a solution to the problem in XML.

XMETAL

It's important for the user interface of an authoring tool for structured documents to feel familiar to someone used to modern word-processing applications. It's also important that the tool have the necessary power. XMetaL is a user-friendly XML editor developed by SoftQuad that

LISTING 1

```
<CHANNEL HREF="http://www.softquad.com/sports.htm"
BASE="http://www.softquad.com/sports/">
<TITLE>Your Sports Center</TITLE>
<LOGO
HREF="http://www.softquad.com/sports/wide_logo.gif" STYLE="IMAGE-WIDE"/>
<LOGO HREF="http://www.softquad.com/sports/logo.gif" STYLE="IMAGE"/>
<LOGO HREF="http://www.softquad.com/sports/icon.gif" STYLE="ICON"/>
...
</CHANNEL>
```

LISTING 2

```
<smil>
<head>
<region id="c1" z-index="5" top="5"
left="5" />
<region id="c2" z-index="6" top="5"
left="5" />
</head>
<body>
<img region="c1" ... />
<anchor z-index="7" top="0" left="0" />

</body>
</smil>
```

LISTING 3

```
<manual>
<title>Setting the Clock</title>
```

```
<process model from="A" model to="B">
<!-- the models that start with "A" or "B"
use this process-->
<step>Open the panel under the TV
screen</step>
<step>Find the button called <button
model="A">Time</button>
<button model="B">Set</button></step>
<step>The buttons next to this button
have up and down-arrows on them. The up-
arrow increases the number on the
display; the down-arrow decreases it.
Hold down the
<button model="A">Time</button><button
model="B">Set</button>
button while you press the up- or
down-arrow to set the time.</step>
</process>
<process model from="C">
<!-- all other models -->
<step>Find the remote control.</step>
<step>Press the <button>Menu</button>
button <var model="CA">once</var>
<var model="other">twice</var>, to get
to the set-up menu.</step>
<step>Choose "Clock".</step>
<step>Press up-arrow on remote to
increase the time or the down-arrow to
decrease the time.</step>
<step>Press <button>Set</button> to
finish.</step>
</process>
</manual>
```



combines a user interface modeled on common word-processing applications with an XML parser, XSLT processor and a DOM interface.

The Softland Airlines demo described above is one example of the type of application made possible by the HoTMetaL Application Server. HMAPPs can access data from flat files, xBase database files and ODBC sources, and from HTML and XML documents. It can produce HTML, XML and client-side script, which is delivered to the browser, and the language used to script the Web page is based on XML. This means that it is familiar to people who have written Web pages in a way that scripting languages often aren't.

Conclusion: Why the World Is So Excited About XML

Just a few years ago, XML was but a dream in the hearts of a few visionaries who, in May 1996, decided to form a group to define a simplified subset of an ISO standard called SGML, creating a universal data and document metalanguage for the Web. With over 10 years' experience with SGML, the group knew what was useful and what could be left out without unduly losing functionality. The result is XML, a simple, powerful way of describing the formats of both data and documents.

XML has come along at just the right moment in the lifetime of the Internet. The explosive growth of the Web has resulted in people rapidly pushing HTML to its limits to produce sites that are pretty but isolated and offer only coarse-grained personalization. The next wave of Web development will be sites that are not only data-driven but offer enhanced services built on data integrated from multiple sources and whose intelligence is carried through right to the browser for individual exploration and manipulation. It is XML's ability to richly describe data and preserve its meaning through all the steps of a delivery and processing chain that makes this next wave possible.

What can you do to be ready for this future? First, because so many benefits of XML are tied to its predictable structure, be aware that XML applications are going to be more demanding than HTML with regard to having correctly structured documents. You should be thinking about combining XML and HTML. To do so, you'll need your HTML to be well

formed and to follow "good practices," such as putting quotes around attribute values and using case consistently. Is this hard to achieve? Not if you use the right tools. (You can hardly expect me to refrain from mentioning HoTMetaL as being arguably foremost among these!)

Second, recognize that the trend is toward Web applications with an increase in processing on the client side, but even more so on the server side with application servers. Application servers that can both consume and produce XML are going to be necessary in the coming XML world.

Last, as we've shown, XML is used for many different things; it's an extremely flexible format with one basic concept, that of marking up your data with information saying what the data is. This means that XML finds uses as:

- A data description format
- A data interchange format
- A data storage format
- A document storage format
- A trigger for specialized processing

The examples in this article reflect different aspects of this flexibility. The most rewarding applications of XML will no doubt be those that combine all those aspects.

XML Reference

SoftQuad XML resource page: www.softquad.com 

AUTHOR BIO

Bruce Sharpe joined SoftQuad, Inc., in 1996 and has been responsible for product development for HTML, XML and SGML authoring and delivery. Bruce has led several software research and development teams since he obtained his Ph.D. in mathematics from the University of British Columbia in 1984.

B.SHARPE@SQWEST.BC.CA

Call for Writers...

XML-Journal is looking for qualified contributors who can offer insights into XML and related technologies in the following areas:

- **XML Tutorials:** Introductory tutorials with plenty of code samples for novices and other people new to XML and data modeling
- **Tips & Tricks:** Techniques gained through real-world experience in using XML
- **Microsoft's XML strategies:** Inside information on Microsoft's direction regarding XML and articles covering BizTalk
- **Humor:** An informal column on e-business and XML that adds some entertainment value to the magazine. The focus should be limited to related technologies. If you'd like to submit cartoons, we can consider that a possibility too.

Contact
Cheryl VanSise,
Managing Editor
cheryl@sys-con.com



Get Your Own
Subscription to the
Finest Technical Journals
in the Industry!

1-800-513-7111
www.sys-con.com



The technology du jour may be XML, but what's needed is to cut through the growing infoglut that's surrounding it

XML and Databases: 'There's Too Much Confusion!'

In the past year I've encountered much confusion about XML and the role of databases. If you're among the confused, don't feel bad. XML is the technology du jour and there's a steady stream of new information. Vendors are competing for mindshare, so their white papers and product literature add to the confusion about databases and XML servers. In addition, many developers simply don't understand databases. To help reduce the confusion, this article examines some database concepts and looks at solutions related to XML, data access, databases and data integration.

XML enables us to define vocabularies for a wide spectrum of applications. Most XML applications fall into one of two categories: publishing and data integration. For convenience, assume an application that sends content to a browser for a human audience is a publishing application. A software application that emits XML for not-for-human consumption is using XML for data integration. Neither type of application is immune to problems related to version control, authentication, authorization, searching and large volumes of data.

Managing and searching XML data is harder when content exceeds a small number of files. Version control for collaboration and security for applications having sensitive information are problems if you have multiple authors or users. Another issue is the need for sophisticated indexing and searching techniques. These are required to search a large collection of documents or to execute content-based queries. A content-based query can match text in an XML document, patterns in an image, a location in an area, or all three.

Data volume can also be a problem. Publishing projects deliver pages to a

browser but they may need to search large data stores. The (U.S.) National Library of Medicine's MEDLINE databases include over 10 million bibliographic citations being converted to

XML. Using an XML-enabled Web browser, you'll be able to search the world's largest medical literature database for information such as the citations shown in Figure 1.

Server-to-Server Messaging

Data integration is a server-to-server scenario instead of server-to-browser. It involves an exchange of XML as data streams intended for machine process-



FIGURE 1

Publishing with XML can involve very large databases. The U.S. National Library of Medicine is converting 10 million bibliographic citations to XML for Web access.

ing, not individual documents to be read by a person. Servers exchanging B2B messages are an example of data integration. They can, for example, use XML to transmit and confirm purchase orders. Other examples of XML for data integration include feeding metadata to a data warehouse, content syndication and exchanging information between disparate databases.

B2B applications often extract data from databases before sending an XML document, and store information in a database after receiving one. In addition, B2B messaging may require a database as a persistent store. This provides a safety net when an application deals with a large volume of messages or supports message queuing.

The XML world is not a narrow niche, and being an XML developer today is somewhat like being a handyman. Whether you use XML for publishing or data integration, you need a mix of skills and tools. Knowledge of databases and data access techniques can be a useful addition to your skillset. If your role is that of a system architect, you'll need other skills. The system architect should understand how XML relates to objects and components, integration servers, Java, portals and messaging middleware. That's too much to cover in a single article, so I'll focus on databases and data access.

Database Concepts

People who come to XML from publishing or designing Web sites might not have database development experience (design or programming). Because they don't understand how databases work, they don't know how to use a database for problems such as concurrency in collaborative environments. They amass a collection of XML files because they don't know how to store XML in a database.

To effectively use databases for XML projects, you need to understand certain fundamentals. First, a database management system (DBMS) provides a consistent means of accessing data, particularly data that's shared among applications and multiple users. Databases come in more than one flavor. They don't all have the same physical organization or the same logical model for accessing information. Since the 1970s, the industry has published standards for databases that conform to three different logical models – linked lists, sets and objects. Objects are data plus methods that operate on that data.

Databases aren't a passive container

like an XML text file. They're flexible because you can store behavior that suits your application needs. You can store XML documents in their entirety, or decompose them. In either case you can embed logic and rules in a database to help produce XML applications having consistent behavior. (Uniform rules and consistent behavior are the rationale for creating XML schema repositories such as BizTalk.org and xml.org.)

Physical and Logical Models

Successive generations of database technology produced different physical and logical models for data. The nodes or tree structure of an XML document is a hierarchical data model that first appeared in DBMS products in the 1960s. Because this type of database dates back to the mainframe era, you can still find legacy applications using a hierarchical DBMS. Some developers see XML as an excellent solution for data integration with these legacy databases. If you don't have legacy data and are looking for off-the-shelf database technology, you can expect to use relational, object or object-relational databases.

As database models evolved, so did the languages for querying databases. SQL is a query language that's been an international standard since 1986. IBM originally developed SQL as a solution for working with relational data, but there have been several standards updates. SQL is a set-oriented language that provides the ability to treat data as tables, and more recently as objects.

SQL is nonprocedural. You submit SQL statements to a query processor, and it returns a result set containing rows matching your search criteria. Other names for the rows returned by a query are *recordset* and *rowset*. Many popular SQL products use a client/server or distributed processing architecture. The SQL server executes queries sent to it by a client application or another server, such as an XML server. SQL servers enable you to embed stored procedures and other logic in databases that serve multiple clients. A stored procedure is a precompiled SQL script that executes at the server. Because stored procedures offer performance benefits, database vendors such as Microsoft encourage their use for serving up XML from databases.

Object-Relational Databases

The early versions of SQL enabled us to store and retrieve tables of numbers and characters. However, object-rela-

tional technology changed that. With an ORDBMS you can store and retrieve images, video, text and geospatial data. Because XML is structured text, an object-relational database can treat XML data as tables or objects, and store information about a document's structure.

Today IBM, Oracle and Informix offer object-relational DBMS (ORDBMS) products having an SQL interface. Their architecture is extensible so you can support a new type by adding a server plug-in. This is similar to the way you can use a Web browser plug-in to read Adobe Acrobat documents. Browser and SQL servers share another common trait. They include a Java library and embed the Java Virtual Machine. A browser uses the JVM to execute Java applets downloaded with Web pages, but an SQL server uses the JVM to execute Java classes embedded in databases. This gives developers the capability of writing Java classes for processing documents and storing the classes in the same database as XML data, DTDs and even Web pages.

APIs and Schemas

To exploit the combination of XML and databases, a developer needs to understand schemas and APIs (application programming interfaces). Programmers writing XML applications can use different APIs for parsing documents and different APIs for accessing databases. For designing documents and databases, they also use different schema languages.

Schemas are a formal means of describing data and defining rules about it. SQL includes a Data Definition Language (DDL) for specifying database schemas. XML developers can define documents using a Document Type Definition (DTD) or XML schema. Several XML initiatives have been related to schemas, including Resource Description Framework (RDF), Document Content Description (DCD), XML Data and XML Data-Reduced (XDR). The W3C XML Schema Working Group has also developed a two-part draft specification for a schema definition language (see "XML References" at the end of this article).

Part 1 explains structures, constraints, attribute groups, model groups, derived type definitions and validation. Part 2 describes how to specify data types used in XML schemas.

SQL schemas and XML schemas aren't divergent concepts. Both enable you to define custom types to encapsulate information and provide abstrac-

tion to simplify programming. All versions of SQL include the ability to define tables with a CREATE TABLE statement. Some versions provide statements for defining objects or object views. Oracle 8.x includes a CREATE TYPE statement to define new types, including object types. Example 1 is an SQL excerpt that shows creation of an Author type for an Oracle database. Example 2 shows what the counterpart XML Schema might be for defining an Author type; it is based on an XML Schema specification that's still a draft as I write this.

EXAMPLE 1: SQL

```
CREATE OR REPLACE TYPE Author AS
OBJECT (
    authorID          VARCHAR2 (5),
    firstName         VARCHAR2(40),
    lastName          VARCHAR2(40),
    middleName        VARCHAR2(30),
    biography         VARCHAR2(512));
```

EXAMPLE 2: XML

```
<Schema
  targetNS="http://burns/books.xsd"
  version="1.0"

  xmlns="http://www.w3.org/1999/XMLSchema" >

  <element name = "Author">
    <type>
      <element name = "AuthorID"
        type="string"/>
      <element name = "FirstName"
        type="string"/>
      <element name = "LastName"
        type="string"/>
      <element name = "MiddleName"
        type="string" minOccurs="0"
        maxOccurs="1"/>
      <element name = "Biography"
        type="string" minOccurs="0"
        maxOccurs="1"/>
    </type>
  </element>
</Schema>
```

There's no one single solution for retrieving information from databases and delivering it as XML. Likewise, there are different solutions for parsing XML data before storing it in a database.

Parsing Documents with DOM and SAX

The W3C published a specification for the Document Object Model for XML documents. There are XML processors for Java and C++ that will parse a document and create in memory a version of its DOM objects. Compa-

nies and individuals have released XML parsers that are freely available on the Web for developers using Java, C++ and other languages. If you're developing XML applications that use SQL servers from IBM, Microsoft and Oracle, you'll find all three provide a DOM-compliant

will permit the creation of tools that read XML schemas and generate Java classes to access XML documents. The rules for document structure and content will be reflected in generated Java code for parsing and validating an XML document.

The XML world is not a narrow niche, and being an XML developer today is somewhat like being a handyman

parser. Microsoft's parser exposes Component Object Model (COM) interfaces to programs and scripts. IBM offers parsers for Java and C++ and Oracle offers parsers for Java, C/C++ and PL/SQL.

DOM parsers consume more memory as document size increases. This makes DOM unsuitable for applications that require large XML data streams. The Simple API for XML (SAX) is an event-driven API. A SAX parser works through an XML stream and generates an alert for events such as the start of an element. SAX programs can run in less memory than a corresponding DOM program because they do not build an in-memory structure for mapping the document to objects. This makes SAX a better choice for Java classes stored in databases. Both APIs are undergoing revision so you'll be hearing about DOM2 and SAX2.

XML Data Bindings for Java Programs

Databases can enforce certain restrictions before inserting or changing data, but it's important to validate content before sending it to a database. The current generation of DOM and SAX parsers can use a DTD to validate a document's structure. At this writing the XML Schema specification is imminent, so many DOM and SAX programmers are looking to incorporate schemas in the near future. This will make it easier for a developer to write data validation routines based on using type, structure and constraint information.

For developers working with Java, Sun has been developing XML Data Binding for Java as an alternative to the SAX and DOM APIs. This specification

SQL Data Access

We've seen that developers can use different APIs for processing XML data. Given the widespread adoption of SQL, it should be no surprise that there are multiple APIs for accessing SQL databases. Historically, database vendors developed proprietary APIs and then adopted multidatabase APIs as standards emerged. In 1995 the ISO modified the SQL-92 standard by adding a call-level interface, and Microsoft revised the ODBC (Open Database Connectivity) API to align with that standard. ODBC returns data as result sets that conform to a logical model of rows and columns. Shortly after the introduction of Java, Sun introduced the JDBC API. JDBC enables developers to use Java, SQL and advanced objects such as arrays and disconnected rowsets. ODBC and JDBC are ubiquitous. There are ODBC and JDBC drivers for a variety of databases (see online).

Microsoft has also developed an object-based data access API named ActiveX Data Objects (ADO). ADO can return XML query results as a recordset object and use flat files or streams to make a recordset persistent. ADO's XML persistence mechanism writes a schema section followed by a data section. At present the schema conforms to XDR, but Microsoft will eventually use the XML Schema standard. To support data hierarchies such as XML documents, ADO can operate with a hierarchical recordset. Applications or scripts can save recordset objects directly into DOM objects. Microsoft's Web server technology, Active Server Pages (ASP), also provides support for XML. A program or script can save an XML recordset directly to ASP response and request objects.

IBM

www.ibm.com/developerworks

Different programming and scripting languages require different SQL APIs and XML parsers. A C++ programmer, for example, can use ODBC and an XML parser for C++ to write XML programs that access databases. A Java programmer uses a Java XML parser and JDBC. A developer writing Visual Basic code or VBScript uses the COM interfaces exposed by Microsoft's XML parser and ADO. Because of the variety of programming languages and data-access APIs, the large database companies maintain a multi-API strategy. They also offer tools for developers programming with C++, Java and scripting languages such as VBScript and JavaScript.

Tools

SQL vendors have developed tools to simplify the process of using XML and SQL databases. Oracle, for example, provides an XML Developer Kit that includes a program to read a DTD and generate Java classes for processing that type of document. Oracle also developed a Java servlet (XSQL) that can return query results as XML and transform the XML content to render an HTML page.

ACRONYMS

API:	Application Programming Interface
B2B:	Business to Business
DOM:	Document Object Model
DBMS:	Database Management System
DDL:	Data Definition Language
DTD:	Document Type Definition
ISO:	International Organization for Standards
JVM:	Java Virtual Machine
RDF:	Resource Description Framework
SAX:	Simple API for XML
SQL:	not an acronym!

Informix developed an XML Mapper to enable developers to store and retrieve XML data in Informix databases. XML Mapper works with SQL tables and an XML document template to generate the code for the document's objects. For example, given two tables (Authors and Books) and an XML template containing author information, you can generate a Java class that includes functionality for storing and retrieving XML. The Java code generated by XML Mapper includes getXML and setXML methods. In this scenario the getXML method returns a string containing author data. The setXML method uses a filled-in XML template to insert or update data in the database.

AUTHOR BIO

Ken North writes, consults and develops software. He teaches Expert Series seminars and is the cochair of XML DevCon 2000. His latest book is Database Magic with Ken North (Prentice Hall).

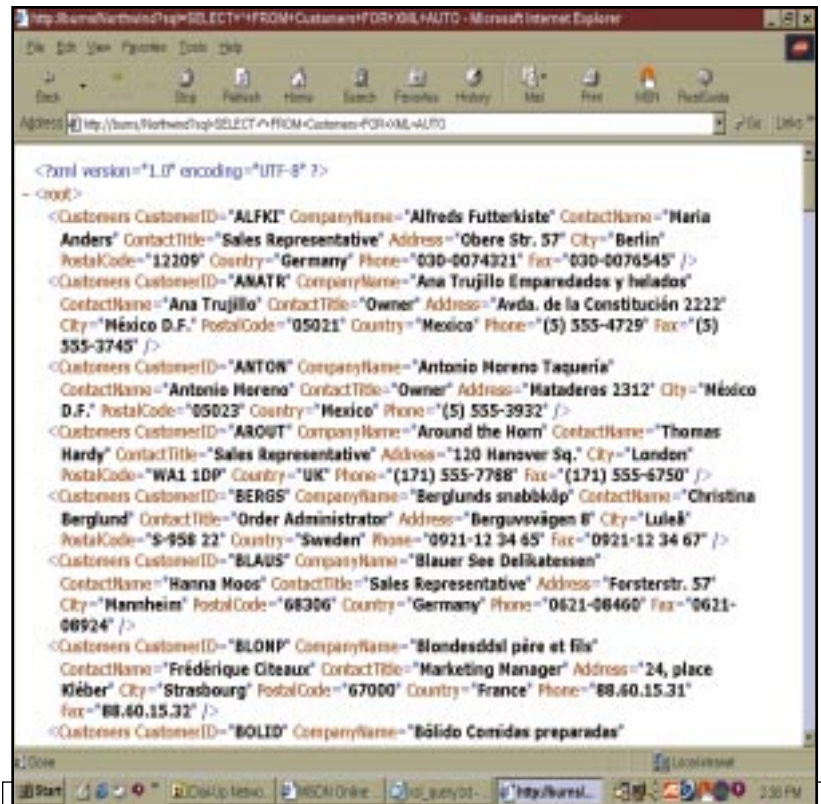


FIGURE 2 The XML Technology Preview for Microsoft SQL Server enables you to specify a query string for searching an SQL database and getting back results formatted as XML.

Informix XML Mapper can also generate a servlet that enables you to take database content and render it for a Web browser.

SQL Server XML Technology Preview

Microsoft is upgrading SQL Server's XML capabilities and, in late 1999, it released an XML Technology Preview for SQL Server. The preview includes a library (SQLXML.DLL) that enables a developer to use keywords in SQL queries to return results as XML. Version 1.0.0.9 of SQLXML works with an ODBC connection to SQL Server. Future releases are likely to include OLE DB connections to Oracle and other databases. Figure 2 is an example of the XML returned by SQLXML when you query the Northwind database.

The Big Picture

This article discussed some aspects of the confusion about XML and databases. The tools and products discussed here aren't the whole picture of XML and data access. Besides the SQL database vendors, other companies have developed solutions for storing and serving XML documents. These include Bluestone Software, eXcelon (formerly Object Design), POET Software, and

Software AG. Space doesn't permit me to discuss them in this article, but you can browse the URLs below for more information. ☛

XML References

1. **XML Schema Part 1: Structures**
www.w3.org/TR/1999/WD-xmlschema-1-19991217/
2. **XML Schema Part 2: Datatypes**
www.w3.org/TR/1999/WD-xmlschema-2-19991217/
3. **JDBC Drivers**
http://ourworld.compuserve.com/homepages/Ken_North/jdbcvend.htm
4. **ODBC Drivers**
http://ourworld.compuserve.com/homepages/Ken_North/odbcvend.htm
5. **Bluestone XML-Suite**
www.bluestone.com
6. **Software AG**
www.softwareag.com
7. **eXcelon**
www.odi.com
8. **Informix Software**
www.informix.com
9. **Oracle**
www.oracle.com
10. **Microsoft**
msdn.microsoft.com/xml

KENNORTH@EMAIL.MSN.COM

SYS-CON Publications

www.sys-con.com

Strategies for a Flourishing Future

Interview...

with **JEREMY ALLAIRE**

COFOUNDER AND CTO OF ALLAIRE CORPORATION, INC.



XML-J: Recently, Allaire has been in the acquisition and partnership mode. Can you give us a brief history of the events over the last couple of years and the rationale behind these decisions?

Allaire: One of the primary reasons for going public over a year ago was to create a base of assets that could be leveraged into mergers and acquisitions. We saw a number of areas for expansion for our platform, all of which underlined a long-term strategy for becoming one of the leading Internet business platforms. In 1999 we undertook three acquisitions. The first, BrightTiger Technologies, makers of advanced scalability and Web systems management technology, underpinned our efforts to bolster our Enterprise-level application server offering. The last two, LiveSoftware and Valto Systems, helped to accelerate Allaire's entry into the Java server marketplace, and have formed the foundation for our next-generation products. Both LiveSoftware and Valto were 100% focused on pure, standards-based Java architectures, including JSP, Servlets, EJB, JTA and JMS. Unlike many other players in the Java server field who were carrying forward their own proprietary Java offerings, we saw that the end of 1999 was the time for a pure-play, standards offering.

XML-J: It seems that this rapid evolution has confused the marketplace. Is there a uniform message Allaire wants to send out to the Web community?

Allaire: A lot has happened with Allaire in the last year. We've evolved from being a leading provider of tools and application servers to supplying a comprehensive Internet software platform covering core server infrastructure, packaged applications and development and productivity tools. If there is a uniform message for the Web community, it's that Allaire intends to be a dominant provider of Internet software platforms, enabling any organization to successfully build their business on the Web. We intend to provide top-to-bottom platform infrastructure, as well as a

wide range of horizontal packaged applications necessary to running an Internet business.

XML-J: Nowadays, in order to get wide acceptance in the computer industry, companies try to adhere to technology standards. Does Allaire plan to standardize any of its technologies so as to get a larger audience?

Allaire: Broad adoption of any platform requires a unique combination of proprietary innovation and open standards. This has certainly been the case in the Internet world. For Allaire, this means building and supplying infrastructure based on Internet and industry standards, innovating beyond standards in territories that aren't developed, and in turn collaborating to ensure that that innovation eventually contributes to open standards efforts. Our efforts in the Servlet and JSP community are indicative of our approach. The JRun team has consistently extended what's possible with server-side Java, and has aggressively contributed that work to the Sun Community Process. Allaire is committed to a similar effort with XML protocols.

XML-J: What does your roadmap for the coming year look like?

Allaire: We've got a lot of things in the works. First off, we're shipping JRun 3.0 – a comprehensive Java application-server offering, including full support for J2EE standards and support for distributed transactions and message queuing – as well as JRun Studio, our first Java-focused IDE product. Later in the year we expect to ship Spectra 2.0, which will add new modules for customer intelligence, personalization and merchandising, as well as expanding the core capabilities of the 1.0 offering. We'll also be delivering Tron, the code name for a new B2B integration server product based on XML middleware. Into 2001 we're planning to deliver next-generation ver-

sions of ColdFusion and JRun through an integrated server code-named Pharaoh, as well as a new offering code-named Harvest, providing comprehensive Web systems management capabilities for managing large farms of Web application servers.

XML-J: What are your thoughts on XML as a standard technology? What do you think are its strengths and weaknesses? Is it truly the epitome of data formatting and transport?

Allaire: We're obviously very bullish on XML as the core infrastructure for the Web and have anticipated its arrival for years. Like other original Web innovations such as HTML and HTTP, XML is extremely simple. For many people, it's so simple that it's a "nontechnology" – like ASCII. But the reality is that because of XML's simplicity and its explicit design for the requirements of Internet applications infrastructure, it will flourish. I think its applications are pretty well understood at this point: structured data storage, structured data exchange and as a framework for developing richer Internet protocols on top of base infrastructure such as HTTP and SMTP.

XML-J: How does Allaire feel about XML and its impact on e-business?

Allaire: It's clearly very radical. Until XML, the principal use and role of Web applications was delivering applications to end users. Web servers delivered applications to end users in Web browsers. With XML, all of a sudden the promise of network effects, of business-to-business integration and of the distributed Web are fueling radical new business models and transforming the "Internet economy." While behind XML there will always be an application platform executing logic and integrating data, XML is what enables the world of transparent information exchange.

XML-J: What is WDDX and how does it relate to XML?

Allaire: One of the most obvious uses of XML is as a middleware layer for the exchange of complex structured data used by Internet programming languages. WDDX, or the Web Distributed Data Exchange, is a simple protocol for the serialization and exchange of programming language data over the Web. It was designed to make simple, lightweight distributed objects possible over the Internet, using any Internet programming environment. WDDX simplifies the work a developer must undertake to tunnel data between sites and business partners. As a technology, it's in the same class or family as SOAP or XML-RPC. WDDX was introduced in late 1998 as an open source technology, and well over 10,000 developers have used it in a variety of applications. Modules are available for Perl, Python, Java, PHP, ColdFusion, COM/ASP and JavaScript. Developers interested in learning more can download the WDDX SDK from www.Wddx.org.

XML-J: CFML is your proprietary markup language. What's the relationship between CFML and XML?

Allaire: CFML is a programming language, while XML is a format for encapsulating data. CFML was designed to

provide a highly flexible, highly productive tag-based programming environment for dynamic content generation using HTML and XML. CFML can easily generate XML, or any other text format for that matter, and also has object scripting functions to do things like parse XML or transform it using XSLT. CFML is an equal cousin to languages like JavaScript and Perl but was designed from the ground up for server-side applications focused on HTML and XML delivery.

Because XML is principally designed to encapsulate and store data, it has a very strict set of syntax rules that would make it quite cumbersome if used to create a standard scripting language. Syntax for things like variable assignment and expression evaluation don't translate well to pure tags. Declarative functions, on the other hand, translate very well to tags and this is where CFML is used to encapsulate complex components and functions.

XML-J: Is WDDX a consortium standard? Will it ever be one?

Allaire: WDDX is an open source technology, driven through a reference standard implemented by Wddx.org. We think this is a model for de facto standardization that will continue to evolve in the Internet era. Because WDDX has potential applicability to next-generation XML protocol work, the WDDX community is actively engaged in discussions around Web- distributed objects and XML messaging protocols – and we expect to see a lot of this work show up in formal standards bodies such as the W3C and the IETF.

XML-J: Are you planning to use XML for any B2B application integration?

Allaire: This has been a core technology and product focus for the company over the past year, and will become a deeper focus in coming years. The first and obvious mechanism for using XML for B2B integration stems from work we've done in our application servers. All of these supply native XML and XSLT processors, which combined with all of the connectivity, integration and data processing services of the application server provide a great foundation for XML integration. Additionally we introduced WDDX, a higher-level XML framework for distributed Web applications. This ships as a native component of both ColdFusion and JRun. So that's the app server side of the house.

With the introduction of Allaire Spectra, we provide a cutting-edge module for business syndication, allowing a customer to expose both data and services or APIs for integration with other businesses over HTTP. This is really our first-generation take at a B2B framework.

XML-J: How does your XML direction fit into Allaire's overall strategy for enabling distributed applications?

Allaire: It's the central technology for our distributed application infrastructure. We strongly believe that XML protocols for distributed objects and messaging will form the glue that ties together Internet applications.

XML-J: What is Spectra?

Allaire: Spectra is Allaire's packaged applications suite,

focused on packaged modules for content management, e-commerce and customer relationship management. Spectra sits on top of ColdFusion, and can be customized and implemented using our tools.

XML-J: What kind of XML support does Spectra include?

Allaire: Because Spectra was designed from scratch in 1999, we had an opportunity to really think about a new architecture for XML and Internet-centric applications, and this was baked into the core of the system. Using Spectra, a customer is required to develop what we call a Site Object Model. We then use a hybrid XML-based object-relational schema to provide object persistence for site assets. Essentially, the developer gets XML-based content storage for free. At runtime a developer can access the XML object structure and use XSLT or CFML (bundled in the application server) to transform this into custom formats, HTML, and so on.

More exciting is the Syndication framework included with Spectra. This includes both Content Syndication (e.g., publish/subscribe data exchange via standard Internet protocols) and Application Syndication. Both of these are anchored in XML and provide a foundation for affiliate networks and B2B applications.

XML-J: Is Allaire planning to supply technologies for enabling other markets such as CRM, ERM and wireless?

Allaire: Absolutely. As your readers may know, we just acquired OpenSesame, a leading provider of customer profiling, analysis and personalization capabilities. OS will form the core of a Spectra module focused on improving customer relationships over the Internet. On the wireless front, Allaire and Phone.com have been collaborating on developer technology for the "wireless Web" for the past three years. In December we announced the availability of a WAP Toolkit for ColdFusion, and have included this in our visual tools, and it's also included as a native component in Phone.com's UPSDK. Also, because of the XML foundation in Spectra's content management system, it makes a great foundation for delivering Web sites simultaneously to browsers and WAP browsers. We're very committed to making this a core part of all our servers and applications going forward.

XML-J: With your recent technology directions it seems that you're making a play for the B2B market. Will this be as an enabler or as a provider of services? Where's XML going to fit into Allaire's B2B strategy?

Allaire: As already noted, we've made significant headway in providing B2B-oriented functions in both our application servers and packaged applications. Going forward, we're expanding this significantly. As outlined in our Technology Roadmap, we'll be releasing a next-generation B2B Integration Server code-named Tron. This server will provide three major areas of functionality: an XML data transformation engine, a WebORB for application syndication and remote services, and an XML messaging engine. All of these are the core components needed for a robust XML middleware offering. Obviously,

this server will be coupled to and usable within our application server offerings.

XML-J: The term Application Syndication is used frequently in your collateral. How does it differ from content syndication?

Allaire: As we've thought about the B2B universe, we realized that a lot of things converged around site-to-site relationships. Often these relationships are just content sharing, but increasingly they involve the sharing of application data and even application services or APIs. Syndication as a concept really works in describing how companies are forming network alliances or syndicates to create new forms of value in the Internet economy. So for us there are two primary forms of syndication: content syndication, focused on sharing packages of formatted content between Web sites, and application syndication, focused on exposing a remote API to a Web application via an XML-based WebORB.

XML-J: Is this something Allaire defined? How do you leverage XML for application syndication?

Allaire: Others have used the phrase and many have used the concept of syndication. I think it's building up a head of steam.

XML is used for two major purposes in application syndication. First, it's used to package application data via serialized data structures. Second, it's used to provide ORB-like protocol semantics (e.g., interfaces, request-response mechanisms and security context).

XML-J: What contradistinguishes you from other B2B enablers such as WebMethods, BroadVision, Vignette and eXcelon?

Allaire: I think our biggest differentiator is the breadth of our offering. We've built a complete platform spanning core server infrastructure, packaged solutions and development tools. We sell and market our platform on a broad basis, unlike many competitors who price and sell strictly for the top tier of the market. We're leveraging our market strength in application servers to enter key adjacent markets, such as packaged commerce software, B2B integration servers and Web systems management. I think developers will also find that Allaire has a core, very focused Internet-centric culture that pervades all of our products and underlines our technology innovations.

XML-J: How can developers new to XML start taking advantage of Allaire products?

Allaire: With our application servers we provide native support for simple XML processing and transformation. More important, developers new to XML can use Allaire Spectra to easily take advantage, without the pain of developing their own infrastructure from scratch, of the two most central promises of XML: structured data and document storage, and data and application integration.

XML-J: How can our readers get more information on your products? How can they gain access to them?

Allaire: XML Reference

www.allaire.com/tech_roadmap/ 

eXco

www.odi.co

elon

om/excelon



Companies can expand their existing EDI trading systems today using XML

An Approach to Representing EDI in XML

Electronic commerce isn't new – more than 300,000 of the largest companies worldwide exchange electronic trade documents every day. XML-based Internet commerce can't simply ignore this realm. According to Munro's Law, "The value of an e-commerce solution is exponentially proportional to the number of companies that use it."

If we apply this rule, EDI (electronic data interchange) dwarfs the entire combined volume of XML e-commerce – which is why a viable XML-EDI solution is critical to the success of XML e-commerce as a whole.

This column will discuss XEDI, an approach to representing EDI in XML syntax. In this issue I'll provide what I

hope will be a compelling introduction to XEDI. In subsequent issues I'll provide more details on EDI and programming with XEDI.

When the XEDI working group began looking at the problem of EDI to XML, we quickly realized that the challenge was a many-to-many translation (see Figure 1). There were already many

types of EDI. Not only were there two distinct houses called X12 and EDIFACT, but also different release versions of each – and even different industry subsets. There were also many different XML specifications: RosettaNet, cXML, CBL, X12c, OAG and ECO all had different proposed XML representations of trade documents.

The most common approach to solving a many-to-many problem is to break it into two smaller problems: a many-to-one problem and a one-to-many problem. This is the approach the XEDI working group adopted. To solve the XML-EDI challenge we needed a common intermediate step that was also as simple as possible. The result was XEDI.

The Direct Approach

Numerous groups were working on the XML-EDI problem long before the XEDI working group – the ad hoc XML-EDI Group, the X12C Committee, CommerceNet, the SGML Centre and ISIS, to name a few.

Each of the groups took a broadly similar approach: they made a DTD for each business document or transaction set and made elements out of the human-readable metadata. We call this process the direct approach.

It turns out that this approach was actually the hard way to go. Every group that tried it needed to create every transaction set from scratch. As a result, these

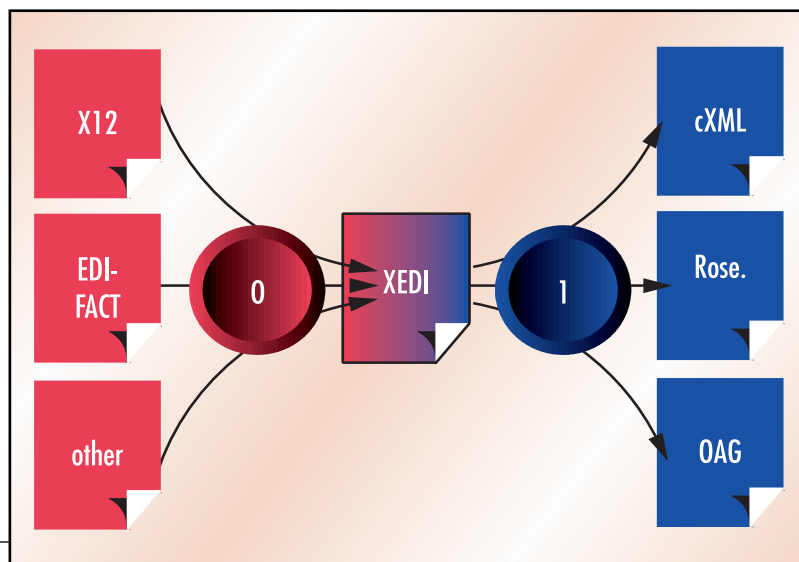


FIGURE 1 XEDI is an intermediate format necessary for many-to-many translation.

groups, even after 18 months of effort, still have only a handful (perhaps 10) of transaction sets completed. XEDI on the other hand now has every transaction set complete in every version of X12 and EDIFACT.

The direct approach creates several problems. It requires a different DTD for every transaction set, which means hundreds of DTDs. It requires a different XML element for every EDI segment and element, which means hundreds of element definitions in every DTD. And it uses the human-readable metadata as the element name, which makes for some very long and unwieldy tag names such as:

```
<ServicePromotionalAllowanceOrChargeInformation/>
```

or

```
<CarrierDetailsSpecialHandlingOrHazardousMaterialsBoth/>
```

More important, these tag names are the *English* versions of the human-readable metadata. By using these versions as the tag name, the direct approach enforces a U.S.-centric DTD – not a viable approach in a global economy.

The direct approach also doesn't address how to incorporate the machine-readable metadata that EDI already has. Most implementations of this approach have dropped the machine-readable metadata altogether. Others have appended the machine code to the tag name, a feature that the document object model (DOM) wasn't designed to support.

The direct approach, overall, leads to a very cumbersome solution.

The Indirect Approach

In contrast, the XEDI working group took an indirect approach. We were designing a DTD that was an intermediate step of a many-to-many translation. In its initially intended use, no one but a handful of developers would ever really see a XEDI document. This freed the XEDI working group to adopt a novel approach – one that turned out to be simpler.

The XEDI DTD is the metamodel of an EDI document. The tags are for transaction set, segment, element, name and value. The same DTD is used by all EDI documents. The XEDI documents are nevertheless self-describing. All the human-readable metadata is stored in the contents of a name element. The machine-readable metadata is stored as an attribute of the elements.

Rather than the DTD, another more manageable set of documents – called the *data dictionary* – describes the hundreds of transaction sets, segments and elements in EDI. The data dictionary files are XML.

Translation

The following is the beginning segment of a typical X12 850 purchase order. Can you find the purchase order number in it?

```
BEG*00*SA*XX-1234**19980301*AE123~
```

An EDI document is simply a series of segments of this sort. The fundamental shortcoming of EDI is that all the human-readable metadata has been stripped away in order to make the documents as compact as possible. The

same segment is shown in Figure 2. The XEDI working group believed that the primary advantage XML could bring to EDI was the reinserting of the human-readable metadata.

The best way to explain XEDI's indirect approach is for me to walk you through the translation of this EDI segment into XEDI:

1. The characters before the first asterisk specify the segment type. Open the BEG.xml file from the data dictionary for this trading partner. We'll assume that the trading partner uses the standard X12 version 4010, which can be found at www.xedi.org/X12/en/4010/segment/BEG.xml.
2. The BEG file specifies the name of this segment. We can generate the following XEDI output that contains the machine-readable metadata as an attribute and the human-readable metadata as the contents of a name element:

```
<segment code="BEG">
  <name>Beginning Segment for Purchase Order</name>
</segment>
```

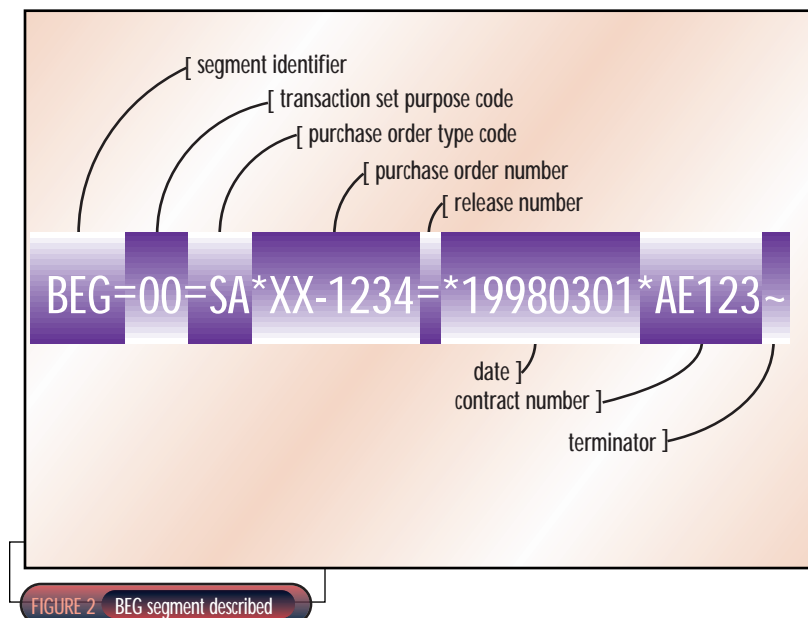
3. The BEG file specifies that the first element is a 353 transaction set purpose code. You'll see the following line in the data dictionary source:

```
<elementRef code="353" req="M"
href="353.xml">
  Transaction Set Purpose Code</elementRef>
```

4. Open the 353.xml to learn what 00 means. EDI is rife with two-digit codes that stand for some business meaning. We find the following line in the 353.xml file:

```
<value code="00">Original</value>
```

5. We can now enter our first element into the XEDI document, shown in Listing 1. The tags describe the metamodel, the attributes contain the machine-readable metadata and the name elements contain the human-readable metadata.
6. According to the BEG file, the second element is a 92. We look in file 92.xml and look up the meaning of "SA." The process works as before and our result is shown in Listing 2.
7. The third element is a 324 purchase order number, but notice that the BEG file has an "element" tag here instead of an "elementRef" tag – indicating that this element doesn't contain two-digit codes for its value. The



content of the EDI document is the value.

```
<element code="324" req="M">Purchase
Order Number
</element>
```

8. The result is shown in Listing 3.
9. The EDI source now has an **, which indicates an empty element. We can insert an empty element or we can insert an element with an empty value.
10. The rest of the elements follow the procedures already described.
11. Notice that the BEG file lists 12 different elements, but the EDI file contains only six. The remaining nonexistent elements are interpreted as being empty. Thus our final result for the segment is as shown in Listing 4.

Step Zero

According to the initial intent of XEDI, no one would ever really see an XEDI document – it was just an intermediate format. EDI would turn into XEDI and then immediately turn into some other XML form such as RosettaNet or cXML.

The XEDI working group stuck to this approach by creating an XSLT stylesheet that transformed XEDI into cXML. We found, though, that in the process we were throwing out a good deal of the information that existed in the XEDI document. The reason was obvious – the semantics of EDI have matured over 20 years, and RosettaNet, cXML and the others simply haven't had the opportunity to mature.

The XEDI working group believes that in time XML-based trading standards will meet and eventually exceed existing EDI standards. We believe that this could happen as quickly as in the next 18–36 months. At that point XEDI can easily map to these standards and serve as the intermediate format it was designed to be. In the meantime XEDI provides companies with the ability to expand their existing EDI trading systems today using XML.

XEDI serves as an intermediate step in an evolution of electronic commerce. It doesn't try to achieve all desirable characteristics such as object-oriented EDI or business-process EDI. It does, however, provide a significant and immediate step forward toward these objectives. It's in this spirit that the X12 SITG committee has called XEDI "Step Zero."

Looking Forward

In the space available to me I've tried to provide as detailed and compelling an introduction to XEDI as possible. It's a wide field to cover and I've had to make various assumptions about the background and knowledge possessed by readers. In coming issues I'll discuss in more detail how XEDI applications work and how you integrate applications using XEDI. You're welcome to suggest questions and issues you'd like me to deal with.



References

XEDI: www.xedi.org
 XMLSolutions: www.xmls.com
 ebXML: www.edxml.org
 X12C Committee: www.x12.org
 CommerceNet: www.commerce.net
 XML-EDI group: www.xmledi.org
 SGML Centre: www.sgml.u-net.com/
 ISIS: www.tieke.fi/isis-xmledi/
 Open Application Group (OAG): www.openapplication.org
 RosettaNet: www.rosettanet.org

RICKER@XMLS.COM

LISTING 1

```
<segment code="BEG">
  <name>Beginning Segment for Purchase Order
  </name>
  <element code="353">
    <name>Transaction Set Purpose Code</name>
    <value code="00">Original</value>
  </element>
</segment>
```

LISTING 2

```
<segment code="BEG">
  <name>Beginning Segment for Purchase Order
  </name>
  <element code="353">
    <name>Transaction Set Purpose Code</name>
    <value code="00">Original</value>
  </element>
  <element code="92">
    <name>Purchase Order Type Code</name>
    <value code="SA">Stand-alone Order</value>
  </element>
</segment>
```

LISTING 3

```
<segment code="BEG">
  <name>Beginning Segment for Purchase Order
  </name>
  <element code="353">
    <name>Transaction Set Purpose Code</name>
    <value code="00">Original</value>
  </element>
  <element code="92">
    <name>Purchase Order Type Code</name>
    <value code="SA">Stand-alone Order</value>
  </element>
```

```
<element code="324">
  <name>Purchase Order Number</name>
  <value>XX-1234</value>
</element>
</segment>
```

LISTING 4

```
<segment code="BEG">
  <name>Beginning Segment for Purchase Order
  </name>
  <element code="353">
    <name>Transaction Set Purpose Code</name>
    <value code="00">Original</value>
  </element>
  <element code="92">
    <name>Purchase Order Type Code</name>
    <value code="SA">Stand-alone Order</value>
  </element>
  <element code="324">
    <name>Purchase Order Number</name>
    <value>XX-1234</value>
  </element>
  <element/>
  <element code="373">
    <name>Date</name>
    <value>19980301</value>
  </element>
  <element code="367">
    <name>Contract Number</name>
    <value>AE123</value>
  </element>
</segment>
```



Java One

<http://java.sun.com/javaone/>

BUSINESS TO BUSINESS INTEGRATION WITH TRADING PARTNER AGREEMENTS

As interenterprise integration technologies such as XML increasingly undergird tomorrow's business models, IBM develops tpaML, an XML specification for B2B interactions....

IBM recently submitted a specification for defining and implementing electronic contracts to the Organization for the Advancement of Structured Information Standards (OASIS), a vendor-neutral standards body. The specification, called *tpaML* (Trading Partner Agreement Markup Language), uses XML and was submitted to OASIS for standardization within its XML.org initiative. This article describes why and how electronic trading partner agreement documents in *tpaML* can be used for business-to-business integration.

The new economy is here and trade is being transmogrified in fundamental ways. In particular, we're no longer dependent on the physical production of goods as the primary engine of wealth and jobs. Increasingly, employment and wealth now derives from the new electronically connected world – where what we produce and consume are ideas, information and services.

In this growing shift toward e-commerce, business rules are changing. As traditional industries start to face nontraditional competition, the vertically integrated corporation – in which a single organization spans and controls an entire value chain – is becoming the exception rather than the rule.

In the new economy, market dynamics already dictate a business model that provides for the integration of different partners in a value chain. Using a variety of IT technologies, this model enables highly coordinated trading communities to operate like a “virtual enterprise.”

Technologies to Support the New Economy

In the emerging Web economy, what's needed is an agile enterprise, one that can work more directly with suppliers and customers and respond more rapidly and intelligently to change. Business pressures – margin erosion, channel proliferation, rising customer expectations, time-based competition and faster product commoditization – are placing increased emphasis on how organizations operate and interoperate with other enterprises.

These profound changes in the business environment demand dynamic and flexible integration between partners in the value chain. While new technologies will be needed to enable such integration, they will need to be able to work seamlessly with existing interenterprise business processes such as EDI and leverage investments in existing enterprise application integration (see Figure 1).

IBM is currently developing new technologies designed to support precisely this type of business-to-business (B2B) integration; they constitute a key component of IBM's application framework for e-business. I'm going to discuss these technologies in this article and I'll be looking too at how they can be used with IBM's MQSeries and WebSphere middleware to build robust and flexible integration solutions. Using this approach, both traditional enterprises and Internet-focused dot-coms alike can more fully exploit the exciting business opportunities of the twenty-first century.

What Are the Requirements?

While facilities such as EDI (Electronic Data Interchange) have been successfully providing document interchange electronically between companies and their suppliers for a number of years, their high cost and inflexible structure have proved a longtime barrier to adoption by all but the very largest enterprises. While EDI will continue to evolve, utilizing pervasive networks such as the Internet to reduce costs, complementary technologies are emerging that are able to provide some of the key capabilities necessary to enable dynamic business process integration.

The basis of these technologies is the formulation of:

- A *common language* that can be employed by existing or potential trading partners to specify how they'll interact
- An *electronic contract* that employs this common language in order to define and enforce the interaction protocols with which they'll do business.

While XML-based B2B protocols such as OBI and RosettaNet are beginning to set a standard for business interactions (albeit currently fragmented), IBM has been expanding on this base through the development of flexible electronic documents termed Trading Partner Agreements (TPAs). These TPAs operate within a Business-to-Business Protocol Framework (BPF) that provides a comprehensive toolset for the specification, configuration, customization and execution of electronic contracts between business partners.

Technologies such as XML and TPAs, coupled with advances in middleware and workflow software, provide the key building blocks needed to underpin an electronic B2B integration infrastructure. Supporting the extensible and easy-to-use TPA format with the BPF framework and middleware from IBM's MQSeries and WebSphere families enables dynamic business process integration by providing:

- *Integration of internal processes*, using modifiable “business rules” to route information between the various internal business information systems
- *Secure, reliable and auditable document interchange* between organizations
- *Externalization of appropriate business functions and processes* to suppliers, customers and partners
- Use of a *broad range of standard message formats, transport and business protocols and network connections* with the capability to dynamically connect new trading partners
- Easy-to-use, business-oriented *single point of control* for interactions across an extended or virtual enterprise
- *Extensible open interfaces* with flexible connectors to link to existing applications

INTEGRATION OF INTERNAL PROCESSES

The first step toward B2B integration is successful integration *within* the enterprise. Proven technologies for enterprise application integration include connectors for coupling to databases and ERP systems, message handling including flexible message transformation and routing, and workflow management for coordination and control of business processes. TPA documents and supporting BPF infrastructure, coupled with the services in IBM's MQSeries middleware and WebSphere application servers, enable these integration styles to be extended to operate between businesses and across firewalls.

SECURE, RELIABLE AND AUDITABLE DOCUMENT INTERCHANGE

A critical concern for interactions between business partners is security, particularly across publicly accessible networks. Features such as reliable sender authentication, audit trails, and secure logging and checking for permissible messages are crucial to secure business trad-

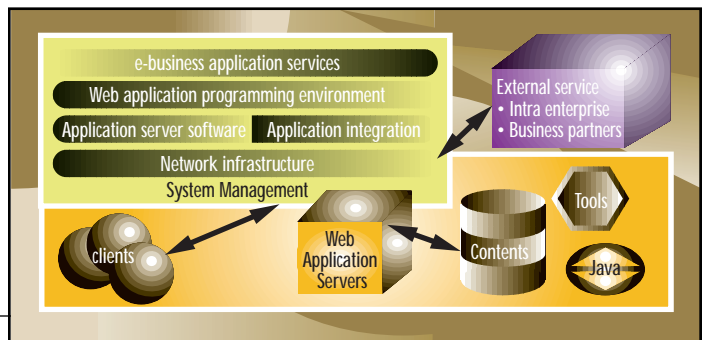


FIGURE 1 Application framework for e-business

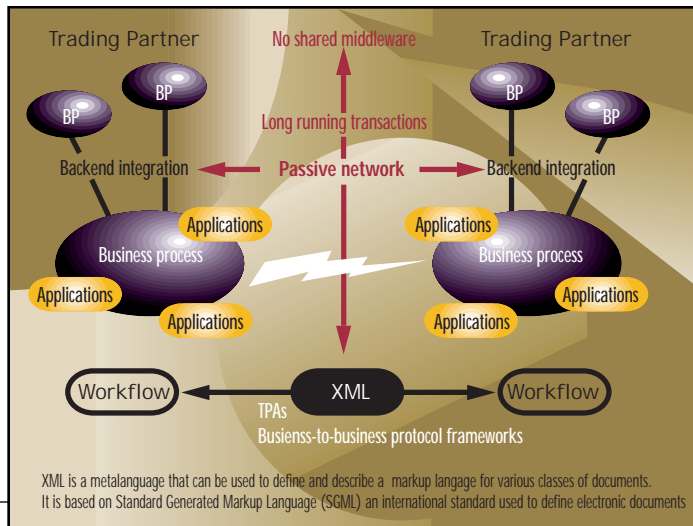


FIGURE 2 Interenterprise integration - issues and solutions

ing. Historically, capabilities such as these have been enabled through value-added networks (VANs), often with EDI interactions. To fully exploit pervasive networks such as the Internet, it's important that these types of security features be maintained.

EXTERNALIZATION OF BUSINESS FUNCTIONS

When a business externalizes automated functions to make them available to business partners, several new issues must be addressed that were less important – or could be solved more simply – within a single enterprise. If the function becomes accessible via a public network, how will it be reached, who'll be allowed to use it, and how will these users identify themselves? What do both parties in the resulting long-running conversation commit to in terms of behavior, responsiveness and recovery from error situations? How can rules of interaction be specified even when the partners have no shared middleware? The TPA is structured to gather the information needed to address these issues and hence provides the basis for middleware that constructs wrappers allowing business functions to be externalized safely.

BROAD CONNECTIVITY

Communications support that enables interaction between potential trading partners – no matter what transport protocol or network they use to connect – is a key requirement for participation in dynamic business value chains. In practice, this means that a B2B infrastructure must support:

- A wide set of common transport protocols including HTTPS (for secure Internet interactions), SMTP (for mail interactions) and reliable messaging such as that provided by IBM's MQSeries
- A wide range of message formats including those based on industry standards for exchanging XML documents
- Existing and emerging B2B protocols including EDI, OBI and RosettaNet
- Transcoding from one message format to another

It must also be possible to merge additional libraries into the middleware base, to support any specialized message formats and protocols that may be needed for specific interactions.

As more dynamic business chains emerge, it will become increasingly important to be able to implement new electronic trading arrangements easily and quickly. This is enabled by providing dynamic connectivity and by leveraging trading partner information in publicly accessible registries and LDAP directories to help identify and locate new potential business partners.

"SINGLE POINT OF CONTROL" AND EXTENSIBILITY

With any extended or virtual enterprise, it becomes necessary to have a consolidated, business-oriented "single point of control." This facility

can help gather together templates for e-business interactions and helps accelerate the connection to new partners and the start of trading interactions with them.

OPEN AND EXTENSIBLE INTERFACES

As dynamic business value chains evolve, the protocol interaction styles in general use will be extended and specialized to encourage increasingly sophisticated interactions between partner businesses. While each business retains specialized *individual* capabilities, businesses are enabled to act *together* as a virtual enterprise to meet the needs of specific industries and business contexts. By providing open interfaces, extensibility points and services to enable the use of XML technology in messages and interfaces, IBM's BPF provides support for these types of extension (see Figure 2).

Trading Partner Agreements

TPAs are XML-based documents used to specify the business interaction between trading partners. A TPA defines how trading partners will interact at the transport, document exchange and business protocol layers. Declarative documents, TPAs specify these roles and responsibilities as a set of attribute value pairs. The document is completed by providing specific attribute information for a particular pair of interacting trading partners.

IBM submitted the specification of tpaML to OASIS for consideration by its XML.org initiative on Jan. 31, 2000. The full text of the proposal is available at www.xml.org/tpaml/tpaspec.pdf.

How Are TPAs Used?

A business uses a TPA document to define the agreed model of interaction with a specific trading partner. The TPA represents a single long-running conversation consisting of a set of related interaction steps, distributed over time but constituting a single Unit of Business (UOB) – a term that can be defined as "a set of actions grouped by the user as associated with achieving a specific business result such as a sale and fulfillment."

Take the example of the conversation between a traveler and travel agent to arrange an itinerary. A TPA is used to define the allowable interactions – that is to say, making reservations, modifying or cancelling them, issuing tickets and confirmations, and making payments. In handling this UOB, the travel agent process will start its own conversations with partner hotel and airline businesses. Each of these subordinate B2B conversations is governed by its own TPA.

One TPA can be used for many independent UOBs between the same trading partners – either serially or concurrently. A single TPA can include definitions for multiple interaction sequences and multiple message formats, any of which can occur in a UOB instantiated from it. Effectively, a TPA acts as the "control center" for all system-mediated interactions with an external entity.

Over time, the accumulation of TPAs can become an effective repository of enterprise interbusiness process descriptions, providing a major tool for enabling process enhancements.

How Are TPA-Specified Interactions Organized?

The TPA simplifies the specification of a B2B interaction by organizing the information into separate functional layers. Dataflow through the runtime layers is governed by specifications in the TPA. This layering provides appropriate abstractions for business dataflow and minimizes the need for specialized coding. Three functional layers are specified in a TPA and supported in underlying BPF runtime processing, as follows:

- **Transport:** Handles the selected transport protocol, network connection and basic security
- **Document Exchange:** Provides document abstraction, including message data mapping, nonrepudiation, time-stamping, logging and auditing
- **Business Protocol Rules and Interface:** Provides message sequence and responsiveness checks, document type and trading partner-specific data handling, together with interface logic to connect to specific local business applications

Geek Cruises

www.geekcruises.com

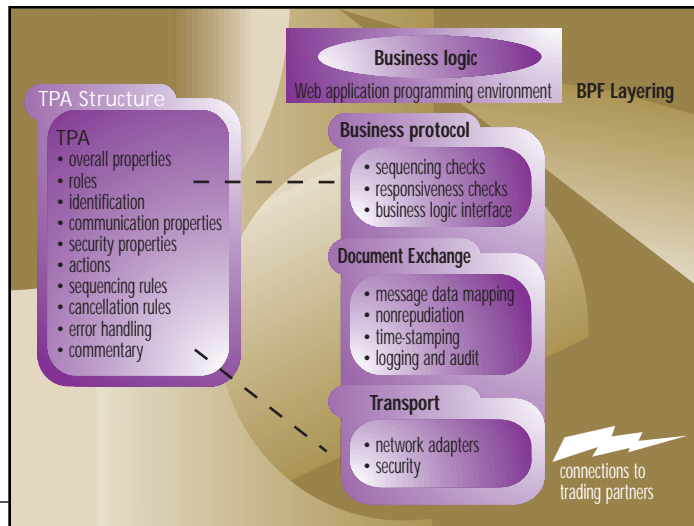


FIGURE 3 TPA and BPF structure

How Are TPAs Structured?

- A TPA document contains the following five categories of information:
- **Overall Properties:** This includes TPA name, starting and possibly ending dates for validity of the agreement and other global parameters.
 - **Role and Identification:** This category identifies the parties to the TPA as logical roles such as “buyer,” “seller,” “airline,” “hotel” and so on. Specific organization names and contact information such as e-mail and postal service addresses are then provided for each role. The allowable actions under a TPA are organized by role, making it easy to modify an existing TPA to specify identical interaction rules but with a different partner. An optional role is that of external arbiter for use in resolving disputes.
 - **Communication and Security:** These attributes specify the communication and interaction protocols to be used. The specification is layered into transport, message handling and higher-level conversational sections. Protocol choices and parameters for security functions are available, such as authentication, certificate handling and nonrepudiation.
 - **Action and Sequencing:** For each identified role this is a menu of the actions that can be performed on request from the partner. A signature is specified for each action defining the parameters and their data types. Sequencing rules specify constraints on the order in which actions can be requested. In the case of the travel agent process interacting with a hotel, example actions would be requests to make a room reservation and subsequently to modify it.
 - **Cancellation and Error Handling:** Cancellation rules indicate whether the result of a completed action can be cancelled, and if so, the constraints under which such a cancellation is permissible. Error-handling rules manage error conditions (such as the maximum waiting time for the response to a request). Commentary text can be added to the TPA to cover other negotiated, but not processed, issues (see Figure 3).

TPA Preparation and Setup with BPF

BPF includes a suite of tools to support the TPA life cycle including preparation, setup and generation of new TPA documents and the generation of interaction processing from them. The tools are used as follows:

- **Authoring Tools** assist the creation of new TPAs. In many cases it may be convenient to construct a new TPA by combining elements from those previously deployed, tailoring the result for the new target interaction. The authoring tools for TPA include modeling and template functions to simplify TPA assembly from preexisting parts.
- **Registration Tools** are used to populate a registration database with interface information identifying the user application logic to be bound into the business-to-business interactions and user-provided helper functions such as specialized parsers, security handlers and encoders.

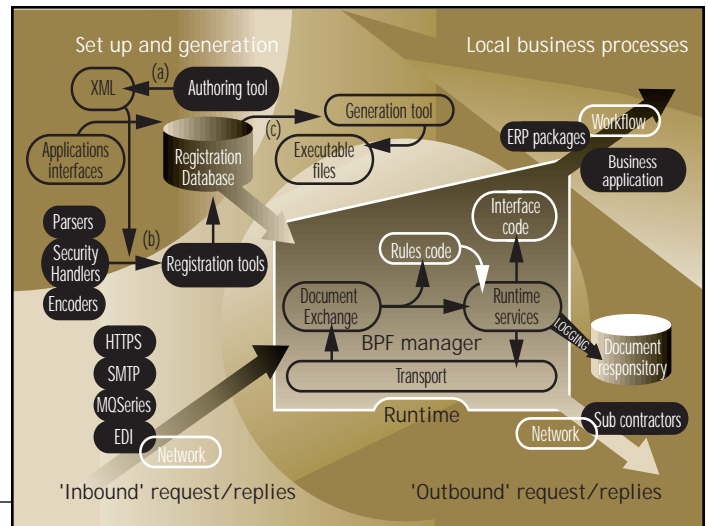


FIGURE 4 BPF setup and runtime operation

- **Code Generation Tools** use the TPA and information from the registration database to generate program objects (executable files) that, together with the runtime BPF, support and implement the required B2B interaction processing.

Hence, starting from a TPA authored with the tools described above, and using a combination of generated and runtime logic, BPF provides a complete implementation of all message and interaction logic needed for a particular trading partner interaction. As TPAs are XML-based, the strong syntactic checking available within advanced XML editors can be used in these tools to diagnose and warn of inconsistencies (see Figure 4).

Runtime Management

At runtime, the BPF Manager uses the data in the registration database and the generated program objects to manage information flow through the system and to control execution of the business rules in the TPA. In addition, functions such as event handling, security services and logging are handled by BPF's runtime services. The BPF-generated code enables interenterprise integration, alleviating the need to develop custom programs to manage networking, business event handling, message processing and sequencing.

Generating interaction code from TPAs allows appropriate security validation and message logging steps to be added systematically. Hence the level of security is adjusted to the communication context of each business partner connection, providing appropriate protection for secure interactions across a publicly accessible network. BPF-generated checks for responsiveness of the partner and valid message sequences improve reliability further.

As noted earlier, to provide an effective single point of consolidation for interactions, a B2B infrastructure needs to provide a broad range of connectivity options. It's also important to provide services for defining, parsing and manipulating XML messages, together with facilities for importing other message libraries and services. This can simplify the mapping of other existing application and message formats both into and out of XML messages.

BPF Runtime Services

When interacting with business partners across open networks such as the Internet, there are likely to be considerable variations in request/response times. In these cases a single business flow requiring responsive interaction with multiple partners, such as in the travel example above, will need to employ parallel concurrent conversations. During these types of conversations, partners may detect unexpected conditions requiring specialized action by the interaction or business logic software. The BPF provides services that map interaction events from message processing into business events that can be meaningfully handled in the business applications. These services enable the coupling

OASIS

www.oasis-open.org

of application processing to appropriate interaction processing, with the ability to control how interaction events are filtered and combined.

Other BPF services provide support for the following five services:

- **Synchronous call requests:** This will allow business applications to make simple synchronous calls requesting an interaction. The framework transparently maps these into asynchronous messaging interactions, preserving the synchronous appearance to the application.
- **Interaction event definitions:** A service to allow the definition of interaction events and event-combining rules that result in applications being "called back" only when a specified set of interaction events or time-outs has occurred.
- **"Callback" rules:** This service indicates which application methods should be invoked when a particular business event occurs.
- **Subordinate conversations:** This provides logic that specifies that a subordinate conversation with a new partner is to be started and considered part of the one currently in progress.
- **Cancellation and compensation:** A cancellation/compensation framework facilitating cleanup when a business interaction has to be cancelled.

Each of these services contributes toward providing a simplified and generic view of external events to internal business applications. Information about the conversation state is available to internal applications to assist in event management. The combination of user-provided interface and rules code with the BPF services mediates between the entirely generated interaction handling support derived from a TPA, and business logic written as a standard application or business workflow.

Using the BPF

Successfully building B2B IT solutions involves a need to understand in detail how to interact and exchange messages with applications and business processes owned by external trading partners. However, the variety of protocols needed for this messaging is innately complex and related to technology rather than to the structure of a particular business process. Defining interaction styles using TPAs, together with BPF runtime services and support from appropriate middleware, provides a rapid, easy-to-use and flexible approach to the development and deployment of B2B solutions.

Utilizing the BPF can improve and accelerate development of new interenterprise e-business solutions in a number of ways, including:

- **Interaction definition simplification:** Structures and simplifies the definition of interactions with trading partners, specifying what each party will expect of the other
- **Choice of connections:** Handles all popular modes of communication to enable interaction and data interchange with any potential trading partner
- **Automated code generation:** Automates the generation of connection and interaction logic enabling rapid commencement of trading interactions with new partners
- **Code reuse:** Allows generic logic within existing business processes and applications to be reused for various partners using different types of interconnection
- **Evolutionary approach:** Enables a smooth transition to broader e-business capabilities by building on any existing EDI or message-based integration capabilities

The Value of TPAs

XML-based TPA documents provide value in that they capture the essential information that trading partners must agree on in order for their applications and business processes to communicate. In integrating the business process of multiple partners, there's value in distinguishing "interaction" information – on which there must be agreement – from more private design and implementation decisions – which can be made autonomously and independently by each participating partner.

TPAs and their associated authoring and registration tools can help speed the process of defining how new trading partner interactions should take place. Since the TPA is a formal XML document, XML tools can provide a valuable level of consistency checking to detect obvious mismatches immediately. In many cases the definition of the interaction style can be further simplified by starting from a sample TPA template as a model, or by basing the TPA on one of the emerging standards for e-business such as Open Buying on the Internet (OBI) or RosettaNet.

Support for all widely used communication modes makes it possible to reach an agreement on interaction rules – and therefore a TPA – with almost any trading partner. This ability to interact with any potential partner provides the opportunity to attract and select e-business partners from the broadest possible marketplace.

The Value of the BPF

Using the BPF tools, a TPA document can be rapidly created to define the interaction between an enterprise's business systems and those of a trading partner. Subsequent automatic code generation from this TPA – combined with runtime services – provides a complete implementation of the required interaction and communication logic. Therefore, within a relatively short space of time, an enterprise can be ready to connect with a particular partner.

When partners have been identified and an agreement reached to conduct e-business – assuming an agreed interaction mode is also ready – conversations can be quickly and easily initiated and business trading begun. This ability to start conducting e-business with new trading partners immediately, without waiting for programming help, provides the ability to attract and select e-business partners in real time. To be able to move quickly and leverage full advantage from these new trading interactions, the programs implementing internal business processes need to drive generic "reusable" B2B interactions.

Choosing new trading partners and starting interactions dynamically also depends on being able to support a range of interaction choices. Since all the necessary protocol checking and messaging logic needed for any one TPA is automatically generated, internal applications issuing higher-level interaction requests will be able to reflect just the business process steps. This separation of partner interaction from business logic flow makes it easier to dynamically add or change partners and helps simplify the logic within business applications. This makes it easier to evolve and extend the function of automated business processes.

BPF helps separate network interaction processing cleanly from business processes. This is more important in B2B integration than within an enterprise because of the greater variety of interaction technologies involved.

Summary

Here's what a tpaML document and a B2B Protocol Framework can provide:

- Tools for authoring or assembling electronic TPA documents and generating interaction code from them
- The runtime system to support the automatically generated interaction code
- Operational tools for managing and monitoring e-business operations with trading partners
- Tools, APIs and services for interfacing existing applications and business processes to the generated logic for interacting with trading partners. ☛

AUTHOR BIO

Francis Parr is a staff member at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, and is currently responsible for technology transfer activities involving joint work between IBM Research and IBM Transaction and Messaging Products in Hursley, UK. He's been involved in e-commerce, messaging and integration products, parallel DB, scalable object technologies and distributed processing. Francis joined IBM Research in 1979.

FNPARR@US.IBM.COM

OMG

www.omg.org

JavaCo

www.javaco.com

n 2000

n2000.com

Permit me to tell you a bedtime story,
albeit a little contrived. It goes like this...

Once upon a time, there were three big-city entrepreneurs: Larry, Moe and Curly

Larry opened a packaging business. He had two types of customers: Buyers and Sellers. The Sellers would drop off different types of merchandise. Larry would store the merchandise in his warehouse, label it for the appropriate Sellers and ship it to them. He was the first to start this kind of business and became a giant in the storage industry.

Moe decided to compete with Larry's business. He noticed that Larry was using inefficient techniques to label and store the merchandise and it was proving difficult to package the merchandise and to transport it to the Seller. His approach was to build a warehouse that was suited to finding the Seller's merchandise more quickly and effectively (or so he thought). He built this warehouse and tried to attract customers. Although his idea was a good one, most customers already had a business relationship with Larry and had a lot of merchandise stored in his warehouse. Moe had to settle for getting the merchandise from Larry's store, restocking it in his own warehouse and then transporting it. This was both inefficient and expensive and didn't solve the basic problem. Moe's business didn't thrive.

Curly adopted a different approach. He built a business that focused more on the packaging, creating customized solutions for the types of merchandise that were being shipped from Larry's warehouse. The merchandise was easily identifiable because of its packaging and it became much easier to ship it to destinations that could route it, because they understood the packaging format.

Moe now saw a new opportunity. He began to create warehouses that could be used for storage while the merchandise was en route. He used his existing business to store the merchandise and was able to leverage Curly's packaging format to route the merchandise. Soon Curly bought out Moe's business and started working in conjunction with Larry. Though Larry wasn't entirely comfortable with the partnership, he wasn't totally opposed to it either as Moe's business no longer directly competed with his own....



Meet JDJ EDITORS AND COLUMNISTS

Attend the biggest Java developer
event of the year and also get a chance
to meet JDJ's editors and columnists



MEETING
September 24-27, 2000
Santa Clara Convention Center
Santa Clara, CA

Sean Rhody Editor-in-Chief, JDJ

Sean is the editor-in-chief of *Java Developer's Journal*.
He is also a principal consultant with Computer Sciences Corporation.



Alan Williamson JDJ/Straight Talking Columnist

Alan, the *Straight Talking* columnist of JDJ, is a well-known Java expert and author of two Java books. A contributor to the Servlet API. Alan is the CEO of n-ary Consulting Ltd, with offices in Scotland, England and Australia.



Ajit Sagar Editor-in-Chief, XML-Journal

Ajit is the founding editor of *XML-Journal* and a well-respected expert in Internet technologies. A Sun-certified Java programmer, he focuses on Web-based e-commerce applications and architectures.



Jason Westra EJB Home Columnist

Jason is the Enterprise JavaBeans columnist of JDJ and a managing partner with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.



ADVERTISER INDEX

ADVERTISER	URL	PH	PG
ACTIVATED INTELLIGENCE	WWW.ACTIVATED.COM	919.678.0300	59
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM		15
EXCELEON	WWW.ODI.COM/EXCELEON	800.706.2509	34,35
GEEK CRUISES	WWW.GEEKCRUISES.COM	650.327.3692	43
IBM	WWW.IBM.COM/DEVELOPERWORKS	800.772.2227	29,68
INFORMATION ARCHITECTS			66,67
INFO SHARK	WWW.INFOSHARK.COM	888-DATASHARK	7
JAVA ONE	HTTP://JAVA.SUN.COM/JAVAONE/	888.886.8769	39
JAVACON 2000	WWW.JAVACON2000.COM	212.251.0006	48-49
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	62-63
OASIS	WWW.OASIS-OPEN.ORG		45
OBJECT MANAGEMENT GROUP	WWW.OMG.ORG	781.444.0404	47
SEQUOIA SOFTWARE	WWW.XMLINDEX.COM	888.820.7917	9
SIMPLEX KNOWLEDGE CO.	WWW.SK.COM	914.620.3700	61
SOFTQUAD	WWW.SOFTQUAD.COM	416.544.9000	2
SOFTWARE AG	WWW.SOFTWAREAG.COM/TAMINO	925.472.4900	13
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM	914-735-7300	25
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM	800.513.7111	25
VSI	WWW.VSI.COM/BREEZE	800.556.4VISI	21
XML DEVCON	WWW.XMLDEVCON2000.COM	212.251.0006	54-55
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM	914-735-7300	31

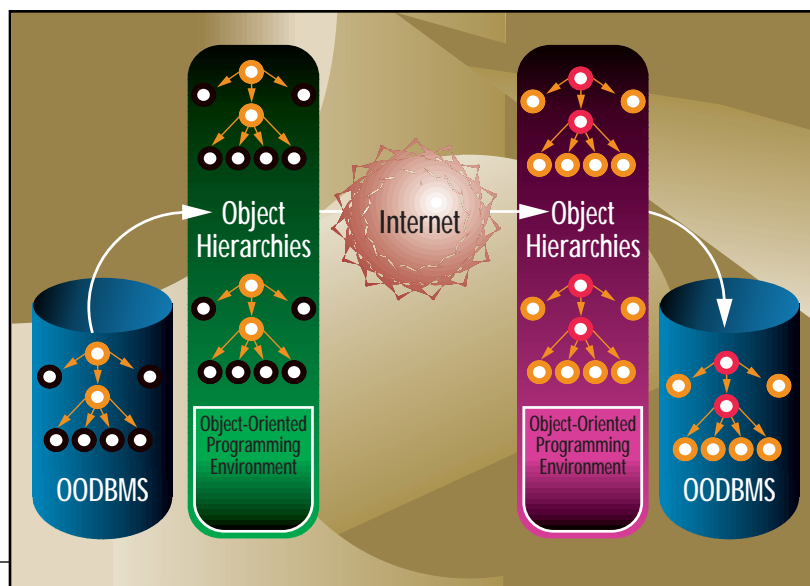


FIGURE 2 Transformation of data using an Object-Oriented Database Management System (OODBMS)

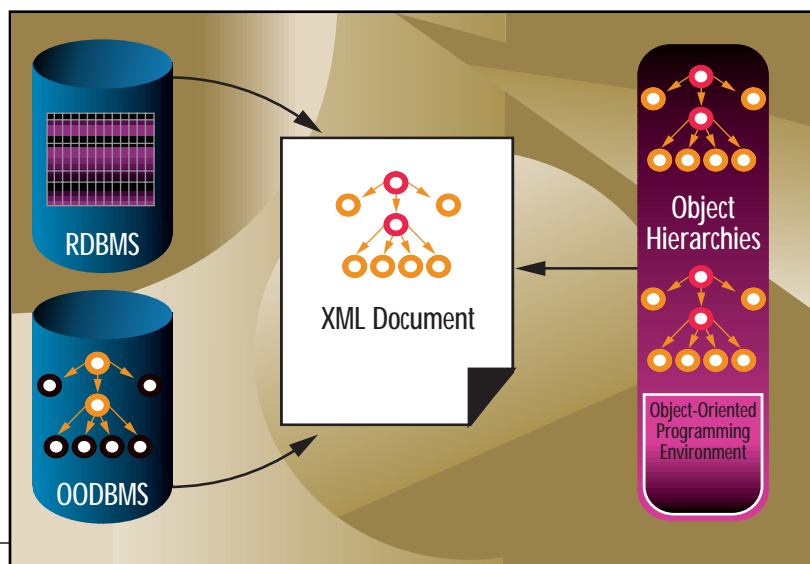


FIGURE 3 Transformation of data using XML

application, manipulated and then stored in a persistent store. The data may also be processed/modified at several other stages in this process. Thus data has to be transformed between several formats at various stages. The data stores are typically RDBMSs. Figure 1 illustrates the transformation of object-oriented data to relational and back.

Because relational databases store data as two-dimensional tables, they are not ideally suited for data manipulation, since data itself may be expressed in the form of complex structures. In object-oriented languages data manipulation is much more efficient, expressing data as objects with rich features – including inheritance, polymorphism and encapsulation – for maintaining relationships

between objects. When the data from an object-oriented world is persisted in a two-dimensional table format, hierarchical relationships between objects aren't preserved, making persistence and recovery of data a complex task.

Does OODBMS Offer a Solution?

Object-Oriented Database Management Systems (OODBMSs) emerged upon the scene in the 1990s, after object orientation had become the name of the game. Object-oriented database management systems are good for manipulating data since they store it in object form, preserving the relationships between the objects. An OODBMS works under the assumption that databases should store data objects that map

directly to the objects defined in the programming language used for writing the application. These systems are based on object models that preserve the hierarchical relationships between objects, retain rich data types and offer a more sophisticated translation from objects to stored data. Figure 2 illustrates the role that object-oriented databases play in object persistence.

However, the bottom line is that data stores for back-office systems are RDBMSs. So a mapping from object to relational and vice versa needs to be created in order to offer persistence capabilities via OODBMSs. The only way object-oriented databases will be able to replace RDBMSs for data persistence and recovery would be if large amounts of data stored in relational systems were migrated to object databases. Moreover, the retrieval and search wouldn't be efficient. In his book, *Com and DCOM* (Wiley, 1997), Roger Sessions discusses this under the heading "The Great Fraud" (pages 211–212). While I don't feel as strongly as Sessions does about the cons of object-oriented databases, I do agree that migrating this data just isn't a feasible option.

Besides, if it were to prove feasible, OODBMSs would seriously threaten the continued existence of RDBMS vendors. In my opinion, OODBMSs made a play for grabbing territory in the persistence game and have settled for *operational* persistence (temporary persistence for data in transit) as opposed to replacing RDBMSs as the *main* persistent stores – mainly because RDBMS vendors have already staked claims in that territory. Besides, relational databases are very good at what they are intended for – storing, searching and retrieving data.

Maybe XML Offers a Solution

XML and databases exist in a relationship of mutual symbiosis. The future of Internet-based business applications is dependent on their ability to exchange information between different data stores or database systems. This in turn is related to the ability of applications to generate XML documents efficiently from data stored in databases. XML allows data to be stored in a format more suited to object-orientation. XML documents are composed of node structures that are made available to object-oriented applications through DOM or SAX parsers. Thus the tools for processing XML documents deal with sets of hierarchical nodes. XML documents can be easily stored as text fields in relational databases. And at the same time XML nodes can be easily parsed into elements

that store the data in object form and maintain the hierarchies of the nodes.

So what's different? you might ask. Data from relational databases still needs to be converted into XML format. Well, relational database vendors are providing the means to query data stored inside flat tables and return the results in the form of XML structures. For example, Oracle – the giant in relational databases – has added features to their 8i product allowing XML documents that have been persisted in relational tables to be queried as XML. This is done by allowing a field in the database to hold an XML document and other fields to hold information about the document. Relational databases can be used to store XML structures, but when the node hierarchy gets deep, persisting relationships involves joins across many tables and becomes very complex and inefficient.

OODBMSs offer a more natural fit for transforming to and from XML. The hierarchy structure in OO databases maps cleanly to the hierarchy in XML documents. On the other hand, relational databases are enhancing their products by integrating with XML servers to provide more sophisticated means of interchanging data in XML format. Remember: the majority of computing data across the world is still stored in RDBMSs.

There's no doubt, however, that XML provides a bridging technology between the two otherwise heterogeneous database systems, OODBMSs and RDBMSs. Since XML data can be extracted from relational databases – and this is a direction strongly supported by the RDBMS lobby – this data can now be distributed into any system that can offer XML transformations (see Figure 3). XML thus provides the bridge between various kinds of persistence systems and makes it unnecessary for object and relational databases to offer some means of direct conversion between one another, i.e., between the two forms of data persistence. All this should hardly be surprising: XML's express purpose is to provide a ubiquitous mechanism for the expression of data structures.

XML Still Needs a Data Store!

It may seem that XML is a great technology for persisting data; however, this isn't its actual purpose. Remember that XML is only a data format. It provides a means to express data in transit between persistent systems. It still needs a database to actually persist the data. This database may be object-oriented or relational.

Consolidation of Two Technologies

An emerging market trend is the consolidation of OODBMS and XML technologies. Object-oriented databases initially made a play for persistent stores for large amounts of data generated from object-oriented processes. This didn't make much headway. Some of the leading OO database vendors are reevaluating their original strategy and jumping on the XML bandwagon. One of these is Object Design, which has now restructured their core products to focus on XML servers. In fact, they changed their name to eXcelon to better represent their market strategy. Although the eXcelon server uses the original OODBMS product, ObjectStore, for persistence, the latter seems to be a smaller factor in the company's direction. Another example is POET's Content Management Suite that is based on their Object Server Suite OODBMS product.

Marking Up

XML, relational databases and object-oriented databases are combining to make an uneasy but inevitable alliance. XML is the unifying technology. Despite the differences, both the RDBMS and OODBMS camps are in agreement that XML is the ubiquitous data formatting technology for today's e-business solutions. It will be interesting to see how these alliances and consolidations shape the future of data persistence in the next couple of years.

vides to them. This is very useful in assessing what exactly you can expect from the book. The author also offers a matrix that provides you with a roadmap for reading the book according to which category of reader you happen to be.

Chapter 1 describes the business problem that XML is designed to solve. The author writes in a very clear, no-nonsense style and makes his points succinctly. Chapters 2 and 3 review XML concepts – brief explanations are provided showing how elements of XML fit into business solutions and simple examples are given to illustrate the various concepts.

Chapter 4 gives an overview of the categories of XML tools in the market today and illustrates how these fit into a business framework. The text is accompanied by clear diagrams that serve to illustrate the author's discussion. Chapters 5–6 discuss how XML plays a role in business processes and enterprise applications. Chapter 7 gives five examples of the categories of XML vendors in the market today.

I found Chapters 4–7 the most useful in the book. The discussions provide comprehensive information that I haven't found elsewhere in so concise a style. There isn't much depth to the discussions but the overviews are very good and give the reader a "big picture" view.

In fewer than 200 pages, Dick covers quite a bit of ground. Minor irritants were the blue sidebars on every page. Most of these merely stated the obvious – such as, "Relationships between docu-

“There's no doubt, however, that XML provides a bridging technology between the two otherwise heterogeneous database systems”

<e-book>

I usually shy away from books that have the word *Manager* anywhere in their title – not because I lack respect for managers but rather because the information presented in such books tends not to be very useful for me.

Kevin Dick's *XML: A Manager's Guide*, published in 1999 by Addison Wesley, turned out to be the exception that proves the rule. This is a great book that provides an excellent overview of how XML plays a role in e-business applications. The author sets the stage for the book in his preface by offering a matrix of the categories of readers who may read the book and the benefits it pro-

ments are important.” I basically ignored these as nothing more than blurbs of information already covered in the text. This is not a book for developers trying to write applications using XML; such readers should try books like *Professional XML* by Wrox Press (published January 2000). But *XML: A Manager's Guide* is a good reference for readers trying to get a handle on what role XML plays in today's business world. You won't get very detailed information but you will get a decent overview of the technology and of how it relates to business enterprise. ☎

AJIT@SYS-CON.COM



Addressing the problems created by the multiple dialects of XML

XML and Data Interchange

We've all heard the proclamations: XML is the language standard that enables seamless data interchange between disparate applications. First the Internet provided the physical connection. Now XML completes this by providing a common language that enables every application to exchange data. Because of this ability, XML will replace HTML as the new lingua franca of the Internet. This certainly sounds like the IT version of nirvana. Unfortunately, reaching nirvana is not that simple.

XML provides an important foundation for representing richly structured data – it simplifies the problem of data interchange. Most companies are building some level of support for XML into their applications. But XML has an oft-overlooked weakness: it does not, by itself, solve the data interchange problem. It provides some common ground rules for how data is represented, but doesn't specify the contents or structure of that data. Because different applications use different content and data structures, there are many different approaches to modeling data in XML. As a result, we have multiple dialects of XML and are forced to try and transform data from one dialect into variant forms so it can be consumed by a disparate array of applications.

Solving the data-transformation problem is critical since data interchange is at the heart of the burgeoning field of business-to-business e-commerce. This article looks at the problem as well as some new technologies and approaches to addressing it.

XML standardizes many things including tagging, hierarchies, nesting, character coding, and more. These standards provide a core level of commonality across all implementations of XML. However, XML is an acronym for eXten-

sible Markup Language, and the key word is *extensibility*, which is what creates the challenge inherent in interchanging data. XML was made extensible so as to support the evolving needs of future applications. In short, developers are able to create their own tags – and the relationships among these tags – to suit their personal needs. This information is captured in schemata called *document type definitions* (DTDs) that can be included with XML or implied, as is the case with well-formed XML. Since XML's standardization by the W3C (World Wide Web Consortium) in 1988, the number of disparate DTDs has increased phenomenally. The problem is that many of these DTDs are competing to solve the same problem. Much like the Tower of Babel, there's been a fragmentation of dialects of XML – represented by DTDs.

Some pundits predict a consolidation of XML dialects. The Internet, they claim, favors universal standards that facilitate interoperability. However, the nature of mankind and early evidence suggest that we can anticipate an increased balkanization of XML. As soon as a few companies form a group to define a standard DTD for their industry, another competing group is formed.

Application vendors also play a role in this fragmentation. Companies benefit in many ways by defining the leading dialect in their market.

- They're perceived as technology leaders by the market; the mantle of technology leadership typically translates into market-share leadership.
- The company that leads such an effort can influence the direction of the standard to support or feature the capabilities of its own application, again increasing its positioning vis-à-vis the competition.
- By having advanced insight into the direction of the standard, these companies can tune their products to exploit the future standard, thereby gaining a time-to-market advantage.

These are strong commercial reasons for application vendors to drive standards efforts. Add to this the unique needs of the various segments of any market and you have a recipe for increased incompatibility. For example, a consortium of companies led by Ariba has created cXML; a consortium led by CommerceOne has created a competing standard, CBL. SAP and Oracle are also creating their own dialects of XML in hopes of making them standards. As a result, XML, which many people viewed as the complete solution for data interchange, is turning into the modern-day Tower of Babel.

Competing dialects of XML can differ in their content, the naming scheme for the tags and their hierarchical struc-

Activated

www.activated.com

ture. One dialect might use the tag <fname> to describe a person's first name. Another might use the tag <first_name>, causing incompatibility. Or one dialect might have separate fields for first name and last name, while a competing dialect groups the first and last names together into a single full-name field. While the naming of the tags and the structure of the tagging scheme are considerations, the most difficult issue to deal with when transforming from one dialect of XML into another is when the basic content of the various dialects of XML differs. If one dialect includes the person's middle name and another doesn't, then there's a bigger problem. In this scenario you can perform a complete transformation only from the dialect with more data to the dialect with less data. But what do you do if you need to transform in the opposite direction? The problem grows when each dialect includes its own unique data. In this scenario you can't perform a complete translation between dialects in either direction.

The complexity of various dialects of XML continues to increase at an amazing pace, exacerbating the problem. Not only are the various dialects becoming more complex as companies try to one-up the competition, but the base functionality of XML is also increasing through the addition of new extensions to XML. An example is the schema-scripting language XML Schema which enables the addition of data-type information to XML. This is a valuable capability because it enables type-aware processing and querying. For example, you can find a product whose expiration date is coming up in the next that. However, this additional data increases the difficulty in translating among various dialects, particularly those that offer different levels of support for these extensions. To make matters worse, XML Schema isn't a standard yet and there are alternative approaches to solving this problem as well.

Fortunately, there is hope.

One of the primary values of XML is the promise of simplified data interchange. This is most critical in business-to-business e-commerce, where companies need to exchange data among various back-office systems that store their data in disparate ways. XML greatly simplifies this process, but the fragmentation of dialects makes the development process very painful. To address this problem, there are two new technologies that solve the dialect transformation problem:

- **XML Stylesheet Transformation (XSLT):** An extension of the XSL draft proposal to the W3C, it provides a means for building standard templates for transformation between dialects, then populating the templates with the data from the source XML.
- **XML Script:** Developed by Decisionware, it's a scripting language that defines the specific transformation procedures and is embedded directly into the XML itself.

XSLT is now at the recommendation stage with the W3C. XML Script, on the other hand, is attempting to become a de facto standard. And, of course, achieving standardization by a standards body doesn't ensure that it will be accepted as a standard. Ultimately, it must succeed in the marketplace to become a standard.

Determining the best approach for transformation also depends on your role in the process. If you're a recipient of XML data, you can fall back on a business solution. You may attempt to enforce your XML standards on the creator of the data. For example, in the buyer-seller relationship the buyer is in a superior position to coerce suppliers to support the buyer's standard form of XML. This relative strength has been demonstrated in the EDI world where companies such as the Big 3 automakers and WalMart have forced their respective suppliers to support their standard data formats. However, the recipient isn't always in such a position and must therefore look at alternative technology-based approaches, such as XSLT and XML Script, to address these needs.

From the sender's perspective, there are two primary options:

1. The sender can generate a standard form of XML, then use tools like XSLT or XML Script to transform this data into the formats required by the recipient.
2. Build an intermediary data store that aggregates the necessary data from various internal sources, then stages it for transformation. The data can be stored as objects in order to maintain its richness and the superior data manipulation functionality offered by a database. For example, by storing the data in a database, the sender can leverage functionality, including rich and standard query capabilities and data type information. These capabilities are coming to XML via XML-Query and XML Schema, but since these technologies aren't a standard

yet, using a standard database can insulate the sender from the risk of building on evolving and nonstandard technologies. By leveraging a staging database, you can programmatically assemble the data into various XML dialects on the fly by simply populating XML templates. The richness of the programming and database languages simplifies this effort.

POET Software was faced with this problem when designing its supplier-side business-to-business e-commerce product, the POET eCatalog Suite (eCS). While data transformation is just one feature of the complete eCatalog solution, it's an important one. In this case the other primary features included data cleansing (so the data is suitable for outside consumption), catalog customization and catalog delivery. To address the data-cleansing requirements, POET needed to stage the data in a separate repository, which also facilitated the ultimate solution for the transformation problem.

POET used its object database, and this offered many advantages. Object databases use a hierarchical structure much like XML, are more extensible than their relational counterparts, offer superior performance when using a complex object model and offer zero administration, making them more embeddable. POET chose to store rich objects instead of native XML. The rich objects are less verbose and more easily searchable. More important, they enabled the product to store a superset of the data required by the various XML dialects necessary for output. From this data eCS generates the actual XML on the fly in the dialect required by the customer. This approach enabled POET to overcome the dialect problem because it could support various dialects, easily evolving as new ones were added or as the current ones evolved. Customers receive their data in the format they need, so the dialect transformation issue was moot.

Of course, if every company generating XML data implemented a flexible data staging solution that translated the data into all of the various formats required by the recipients, the dialect problem would disappear. Unfortunately, as evidenced by the fragmentation of XML dialects, we can't expect all companies to solve their problems the same way. So don't expect the XML dialect problem to go away any time soon. ☛

AUTHOR BIO

Mike Hogan is the vice president of corporate development at POET Software, a B2B e-commerce company. In this role he crafts the company's strategy and establishes and manages strategic partnerships.

MPH@POET.COM

SYS-CON Publications

www.sys-con.com

XML Dev

www.xmldev.com

Con 2000

con2000.com



Debunking the 10 myths making the rounds in the XML-CORBA integration debate

CORBA&XML—HitorMyth?

Is XML a potential competitor to CORBA? Does it represent CORBA's nemesis? Or is it, on the contrary, too lightweight for use in distributed systems? Much heated debate has centered recently on the question of the possible combination of CORBA and XML...

Since opinions on the topic diverge widely, in this article we're going to articulate and debunk some of the more common myths that we've encountered in our work as consultants and product developers.

In the time-honored style of late-night talk show hosts and corporate executives, we've put together a Top 10 list of myths associated with the integration of CORBA and XML. It's worth noting that a lot of these observations also apply when considering the integration of XML with other distributed component systems, such as EJB.

Myth #1: XML IS A COMPREHENSIVE MIDDLEWARE INFRASTRUCTURE, LIKE CORBA

This is perhaps the single most prevalent – and insidious – myth about CORBA and XML. CORBA is a fully featured, stateful, quality-of-service-rich distributed runtime environment; XML is a means of describing document structure. XML is about structure; CORBA is about infrastructure. People talk about building their systems on CORBA in a way that's not meaningful for XML. If you have an XML document and want to transport it somewhere, it's necessary to appeal to other distributed techniques such as CORBA (for more detail on transport issues and misconceptions, see Myth #2).

The part of CORBA that's most directly comparable with XML is its Interface Definition Language (IDL). This allows interfaces to servers to be specified in an implementation-neutral way. However, CORBA implementations then go further and use this information to provide a complete distributed messaging and runtime environment, complete with naming, security, transaction and persistence services.

XML essentially stops at the interface description level, although it does provide a structured and extensible way of describing the payload. Provision of runtime quality of service is left as an exercise for the

system integrator. This is good: XML has a clear design objective and meets it. However, it's important to understand that in order to be part of a fully functional distributed system, XML needs to be combined with real middleware like CORBA.

Myth #2: XML HAS A WIDELY USED TRANSPORT MECHANISM THAT "DOES MESSAGING" AND CAN BE USED TO INTEGRATE WITH CORBA

In fact, there's no single transport or message format of choice for the transmission of XML documents. This has resulted in a number of proposed ad hoc solutions, such as the W3C's XML-RPC and Microsoft's SOAP (Simple Object Access Protocol). Particular transports are often mandated in the context of XML frameworks such as RosettaNet, CommerceNet, OBI, BizTalk, and so on.

Each of these offerings effectively specifies a set of marshaling rules for streaming XML documents across some transport. This transport is usually HTTP, although SMTP is an interesting alternative, leveraging as it does the asynchronous, persistent qualities of the mail system. The problem with any attempt to allow invocations on CORBA systems using XML documents is that some convention is required for encoding CORBA types, references and requests in the XML document that represents the invocation. In effect, what's needed is a DTD for the CORBA protocol, IIOP. Luckily, given the flexibility of XML, it's relatively straightforward to embed CORBA invocation and object information in a suitably structured XML document. Product tools that employ language and code-generation technology are becoming available to automate the process of creating XML documents that map onto CORBA servers and invocations, thus streamlining the process of invoking on CORBA servers from XML-aware environments.

Myth #3: BECAUSE IT'S TEXT, AND COMES IN UNITS OF DOCUMENTS, XML IS TOO INEFFICIENT TO USE IN REAL DISTRIBUTED SYSTEMS

Less efficient doesn't mean inefficient. And with the additional costs associated with XML come commensurate benefits. Though XML text may not be as compact as binary representations, what it provides is portability and readability. As for the concern that documents can be a problematic atomic unit, this can be alleviated by good XML schema design.

In any case, efficiency is a relative notion, not an absolute one. The performance of current XML technology implementations has already proved to be perfectly acceptable in many real applications. As the technology matures, XML processing components will be optimized and current research on compression schemes and fragment streams will begin to bear fruit.

Myth #4: XML IS MORE SUITABLE FOR MOM-BASED ARCHITECTURES THAN FOR ORB-BASED ONES

XML is an excellent format for messages in MOM systems. ORB-based systems can also support messaging, however, as demonstrated by the CORBA Notification Service. In general, using XML affords greater flexibility in data definition, interchange and presentation. This holds true for CORBA implementations no less than for MOM implementations.

XML can be used for messages on CORBA event channels, as data passed into and out of CORBA operations, or as RPC invocations and responses in Web-CORBA gateways. XML also has great potential value for bridging technology domains, and may be an ideal way to tie CORBA and MOM systems together via a common message format.

Myth #5: XSLT IS POWERFUL ENOUGH TO MAKE ARBITRARY EAI-STYLE TRANSFORMATIONS AND THUS NEGATES THE NEED FOR BUSINESS LOGIC IN THE CORBA LAYER

The XML Stylesheet Transformation Language (XSLT) is often presented as

being powerful enough to perform deep transformation of data, which would imply that more complex business logic intervention isn't required as part of an integration solution. This isn't true. XSLT is useful for (and was designed for) applying syntactic transformations as it walks the nodes of a DOM tree. But it doesn't permit the application of semantic rules that would require it to be encoded in some scriptlike or 3GL language – it wasn't designed for this.

For example, one of the authors of this article has used XSLT to convert XML files into IDL (having originally generated the XML from IDL files). This is useful, but wouldn't be suitable if your document processing and transformation required, for example, an operation like "double the value in this data field if today is Thursday."

Myth #6: XML OFFERS NO ADVANTAGES OVER ANYS OR STRUCTS

In fact, the opposite is true. XML can more easily express some involved structures than IDL structs, allowing for complex trees of data. XML can also be easier to change and less brittle than structs, alteration of which requires a cascade of changes down from the original IDL.

XML can be passed to clients with a great degree of client control using client-pull streaming, whereas structs must be passed at once in their entirety. Both Anys and XML (treated as strings or octets) are opaque from the perspective of the IDL in that they conceal typing and semantics. Unlike Anys, however, XML structures are of course fully articulated in XML (possibly via DTDs/schemas).

These gains aside, once data is in XML all the XML advantages of data interchange, presentation flexibility and usage of off-the-shelf components and tools can be leveraged. Beyond this issue, last summer the OMG issued an RFP for representing XML values in IDL – as mentioned in Myth #10, below.

Myth #7: INTEGRATING CORBA WITH OTHER SYSTEMS VIA XML IS COMPLEX AND REQUIRES SPECIALIST KNOWLEDGE OF CORBA IDL AND IMPLEMENTATIONS

In the experience of one of the authors, during his years of product management at a large CORBA vendor, many customers want to maintain two classes of programmers: those who build the back-end systems (often using CORBA) and those who access them (from Visual Basic or, increasingly, Web environments). The latter are often disinclined or unable to learn CORBA, so it's therefore important for them to have ways to access, read and update CORBA objects from within their native programming environment.

A combination of an XML schema or DTD for IIOp, generation of XML from IDL and runtime technology to map between the two makes it possible for Web users to access CORBA servers while having no knowledge whatever of the CORBA programming paradigm. This is good news for all concerned: the constituency of programmers who can access CORBA servers is greatly increased, while the Web jockeys can access and manipulate real back-end business logic without the overhead of learning new programming paradigms.

Myth #8: EXPOSING SYSTEMS VIA XML IS MUCH EASIER THAN EXPOSING THEM VIA IDL INTERFACES

XML is human-readable, and the basics of using XML are quickly learned. CORBA has a steeper learning curve and good IDL takes a certain amount of expertise to write. This has led some to believe that it's much easier to define systems interfaces in XML than in IDL. In fact, defining good XML schemas isn't trivial; it requires just as much thought as traditional OO analysis and design.

Aside from this, neither IDL nor XML has any programmatic value in isolation; both must be supported by plumbing implementations. XML by itself is just data definition, requiring much additional processing programming. Last, an either/or approach is misplaced – there's a lot of value in having IDL interfaces with XML data, as discussed elsewhere in this article.

Myth #9: WHEN EVERY SIGNIFICANT PACKAGED SOFTWARE SYSTEM SUPPORTS XML, THE NEED FOR INTEGRATION INFRASTRUCTURE LIKE CORBA WILL VANISH

Many software vendors – perhaps even most of them – have now announced support for XML and their intention to offer "XML interfaces" to their products. This isn't a particularly meaningful proposition for the most part. Absent the (unlikely) scenario of all vendors agreeing on common DTD formats for their interfaces, we're left in the position of needing first to bridge between the multiple XML document formats, and second to message them around a distributed system once such transformation has occurred.

It seems to us that the effect of this is twofold. For one thing, it erodes the value of "traditional" EAI players like Mercator and NEON, whose value today largely derives from their intimate knowledge of proprietary data formats – inter-XML document transformation in the pure XML world can be largely handled by XSLT. And for another, the increased prevalence of "XML interfaces" to sys-

tems will lessen the interface integration challenge and refocus the problem onto the underlying middleware – for which CORBA remains the premier choice.

Myth #10: THE OMG IS NOT XML-FRIENDLY

Just the reverse is true, as shown by a continuing series of OMG actions. XML was used for the deployment descriptors in the CORBA Components specification, and is the basis for XML Metadata Interchange (XMI), which is central to OMG's Analysis/Design efforts.


XMI provides a way to move UML models between design tools and repositories and is an exemplary use of XML for exchanging data between systems. Currently, much effort is being put into the XML/Value RFP process. The goal of this is to standardize a way of defining XML documents in IDL, thus incorporating XML directly into IDL. The RFP was issued last summer and initial submissions have been made by several groups of companies. To foster such endeavors, OMG works with and has ties to XML-oriented organizations such as OASIS and the W3C itself. In sum, OMG has in fact committed itself to leveraging XML to further enrich and advance CORBA and other OMG standards.

Conclusion

We hope that this discussion has gone beyond merely knocking down the obligatory myth-list straw men, and has dispelled any misapprehensions currently circulating within the development community.

We feel that XML and CORBA are truly complementary technologies. CORBA provides the top middleware infrastructure with an evolved object model, robust and scalable features and services, and a mature product market. To this XML adds a standard portable structured information format and off-the-shelf components and tools.

XML can be used to interoperate with CORBA systems via XML RPC mechanisms or XML data exchange. Having CORBA systems accept and emit XML yields great flexibility in data structure evolution, increased portability of data and good mechanisms for data presentation and transformation. Using XML allows CORBA developers to leverage the ever-growing array of XML-based and XML-compliant products instead of constructing one-off proprietary solutions.

Transcending all myths entirely, combining XML and CORBA is a realistically valuable proposition today. 

 MARK@XENOTROPE.COM

 DAVID.CLARKE@CAPECLEAR.COM

AUTHOR BIOS

Mark Elenko is a senior consultant with Xenotrope (www.xenotrope.com), a distributed systems consultancy in New York. Mark focuses on Java development with technologies such as servlets, CORBA and XML.

David Clarke is a founding director of Cape Clear Software Ltd., leaders in XML product infrastructure. Previously he was a product line director at IONA Technologies, Inc., a leading object middleware vendor.

JDJ S

www.jdjs

Store

store.com

XML NEWS

OASIS and HL7 Exchange Memberships

(Boston, MA, and Ann Arbor, MI) – In a move expected to further XML application and interoperability in the healthcare industry, OASIS and Health Level Seven (HL7) are exchanging sponsor memberships. The reciprocal membership opens communication between the cross-industry efforts of OASIS, which advances XML interoperability through its XML.org industry portal, and the industry-vertical XML development work of HL7, which has made great strides in implementing XML for clinical patient and healthcare services. Both groups are international in their focus.  www.hl7.org www.oasis-open.org

Introducing Stilo XMLDeveloper


(Cardiff, Wales) – UK e-commerce development tools and services specialist Stilo Technology has announced a breakthrough in applications development with the  launch of Stilo XMLDeveloper.

This software delivers huge savings in development time, enabling organizations to bring new e-commerce applications to market faster than ever.  www.stilo.com

RSA and Netfish Form Alliance

(Englewood, CO, and Santa Clara, CA) – RSA Companies and Netfish Technologies are forming a strategic alliance allowing the two companies to deliver a set of Internet-enabled business process integration tools and solutions to the Baan customer base. The joint development  effort has resulted in a “Baan Adapter” that will seamlessly integrate the Netfish XDI system with the mission-critical applications within the suite of Baan BackOffice applications. The companies also announce that RSA Companies has become a certified sales and integration partner of Netfish and will be delivering Netfish products and services to customers of Baan as well as other ERP systems.  www.netfish.com <http://rsacompanies.com>

XML-Based Portal Gains Legacy Data Access via New WRQ-DataChannel Partnership

(New York, NY) – WRQ, Inc., and DataChannel, Inc., announce a partnership designed to provide enterprises investing in e-business with the ability to extend the usability of legacy data in IBM mainframe, AS/400, UNIX and VAX 

host applications using XML as their e-business data exchange platform. Through this partnership, enterprises can use the WRQ Apptrieve application mining solution to selectively expose valuable data in legacy applications for integration into DataChannel's leading XML-based enterprise information portal (EIP).  www.wrq.com www.datachannel.com

The Technical Resource Connection, Inc., and Nova Laboratories Combine Strengths

(Tampa, FL) – The Technical Resource Connection, Inc. (TRC), and Nova Labs announce a partnership that will expand Nova Labs' base of training operations to the Southeast, enabling Nova Labs to use TRC's existing training facilities located at the company's Tampa, Florida, headquarters.  www.trcinc.com www.nova-labs.com

Introducing Unicode 3.0

(Mountain View, CA) – The Unicode Consortium releases Unicode Standard Version 3.0, the software specification that ensures a single, universal way to represent text worldwide. Version 3.0 now supports 49,194 charac-

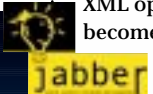
ters, including 31% more ideographs for Japanese, Chinese and Korean markets. Implementation support is greatly expanded, with double the character property data and four times as many technical specifications for supporting implementations.  www.unicode.org

SageMaker Announces SageWave 4.0

(Fairfield, CT) – SageMaker, Inc., a leading provider of Enterprise Information Portals (EIPs), has launched SageWave 4.0. This release includes SageBus, an integration platform based on open standards using Java and XML technology, a new display engine for faster, easier personalization tools, and SageCommunity, a collaboration environment for information sharing.  www.sagemaker.com

Webb Interactive Services Launches Jabber, Inc.

(Denver, CO) – Webb Interactive Services Inc. announces the formation of Jabber, Inc., a subsidiary focused on the commercialization of a revolutionary open source, XML-based instant messaging platform. Webb's role in the integration of this innovative technology will allow the

XML open source platform to become a foundation of advanced commerce and customer service, as well as mobile and enterprise applications.  www.jabber.com

ConSyGen Announces New Mobile Commerce Solution

(Tempe, AZ) – ConSyGen has developed “BizPay,” an Internet merchant software solution that enables businesses to accept the “new” Web-based forms of payment. ConSyGen supplies a method for providing merchants with the full range of accounting, auditing, reconciliation and inventory controls when accepting “e-mail” forms of payment.  www.consygen.com


Pervasive and SYS-CON Offer Trial Subscriptions of Tango Developer's Journal

(Austin, TX, and Pearl River, NY) – Pervasive Software Inc. and SYS-CON Publications, Inc., jointly announce a free trial subscription program for **Tango Developer's Journal (TDJ)**. The first 500 individuals who purchase Tango 2000 products or upgrades from the Pervasive Online Store or who register Tango 2000 products or upgrades will receive a complimentary one-year subscription to **TDJ**.

Published quarterly by SYS-CON Publications, **Tango Developer's Journal**

is the industry's only publication devoted to the subject of end-to-end Tango Web application development and deployment.

The free, onetime **TDJ** trial subscription offer is available to Tango 2000 purchasers in the United States only through September 30, 2000, or while the supply of complimentary promotional subscriptions lasts.

You can go to www.pervasive.com/purchase/ to purchase Tango Development Studio and Tango Application Server products from the Pervasive Online Store.  www.pervasive.com www.sys-con.com



Are You Getting Your Company 'XML-Ready' for 21st-Century Growth?



WRITTEN BY BRUCE SHARPE]

The very fact that you've picked up a copy of *XML-Journal* or are viewing it online at www.XML-Journal.com and are reading this editorial means that my job is half done. Because it shows that you're already paying attention to XML and thinking about how it will change the world of business. But you may be confused about just what it all means to you, and which direction you should take. So let me try to unconfuse you....

First, the buzz about B2B (business-to-business) e-commerce ushering in a new era in the economy isn't just hype, it's for real. We've all heard the projections about the value of online B2B transactions going from billions this year to a volume measured in the trillions within three years. It's hardly an exaggeration to compare the phenomenon with the building of railroads in the 19th century, a development that opened up whole new avenues of commerce and led to decades of unprecedented economic expansion. In the twenty-first century, business giants like GM, Chevron and Sears are leading the way, with infrastructure provided by new e-market makers like Commerce One and Ariba.

Second, despite all the hype about XML, its role in the infrastructure of B2B e-commerce is, if anything, underhyped. XML is without question a key enabling technology. Since almost all e-market makers take it as a given that XML is the backbone of their systems, it's not a matter for debate. It's the right technology at the right time, and it's what everyone is using.

Finally, when you hear that e-markets are the way business is going to be done in the new economy...believe it. The efficiencies and cost savings for buyers are too great to ignore. Cost savings of up to 80% are being reported already. There are benefits for suppliers, too: B2B e-commerce creates new channels with new customers, lowers costs per transaction and offers increased revenue. But the most compelling reason for suppliers to participate is that their customers will demand it. Buyers would like to reduce the number of suppliers they deal with, and the ones they will drop first are those that are not online with them.

Should you wait until the various XML-related standards and schemas that are being discussed have been shaken out and all the dust has settled? In a word, no. Things are moving very quickly in the B2B arena already. But it's nothing compared to the rush to participate that we'll see when the many e-markets – already numbering about 400 now

and predicted to go to 10,000 over the next two years – achieve a critical mass of buyers and suppliers. There really won't be any choice but to participate or be left behind.

So what should you be doing about it? The first step is to investigate how to put your own house in order. You don't need to try to change all your business processes overnight, but you do need to assess where your existing processes need to go. If all your product information is in paper-based brochures only, it's going to be a challenge to leap into an XML-based e-catalog system when the time comes.

As part of your company's XML adoption process, you'll need to act as a driving force. You'll need to ensure that resources are made available and are given the time and the support to learn XML techniques and practices and participate in the growth of the technology.

Don't worry about picking the wrong schema or even implementing one of your own design. Remember that using *any* kind of XML schema is better than not using the technology at all. By imposing a schema on your data, you're putting an order and a structure onto it and it's always an easier task to take an existing structure and convert it to a different schema than it is to impose order over chaos. You can participate in e-markets now and, as the standards evolve, you'll be ready.

As a business you need to be able to distinguish between product data and richer product content. Your customers will want to see more than just part numbers and prices. You need to get the numbers from your product databases, whether they are in Excel spreadsheets or full-blown ERP systems, and put them together with the

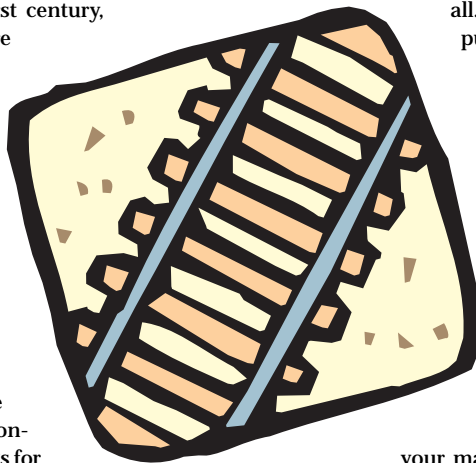
information in the product brochures from your marketing group. You need to create the means of

managing all this content in a way that can be communicated to one or more e-markets and will scale as you add products and customers.

The key point is that you need to be preparing to communicate your organization's products and capabilities to e-markets, which is where your customers are going to be. For all of that you're going to need XML. ☎

AUTHOR BIO

Bruce Sharpe is chief technical officer of SoftQuad Software Inc.



Information Architects

Information Architects

IBM

www.ibm.com/developerworks.com