May 2002 Volume 2 Issue 5

WebServices JOURNAL

.NET J2EE XML

Web Services for Enterprise Application Integration

C++ vs C#

THE FRAMEWORK BEHIND WEB SERVICES INTEGRATION

EAI FOCUS

J2EE-Based Application Servers

Will Web Services Mean the End for EAI?

J2EE, EAI, and Web Services

WebServices JOURNAL
.NET J2EE XML

INTERNATIONAL ADVISORY BOARD
JEREMY ALLAIRE, ANDREW ASTOR, STEVE BENFIELD, DAVE CHAPPELL, PHILIP DESAUTELS, GRAHAM GLASS, TYLER JEWELL, PAUL LIPTON, DAVID LITWACK, NORBERT MIKULA, FRANK MOSS, BOB SUTOR

TECHNICAL ADVISORY BOARD
DAVE HOWARD, JP MORGENTHAL, ANDY ROBERTS, DAVID RUSSELL, AJIT SAGAR, SIMEON SIMEONOV, RICHARD SOLEY

EDITOR-IN-CHIEF: SEAN RHODY
EDITORIAL DIRECTOR: JEREMY GEELAN
INDUSTRY EDITOR: NORBERT MIKULA
PRODUCT REVIEW EDITOR: JOE MITCHKO
.NET EDITOR: DAVE RADER
EXECUTIVE EDITOR: GAIL SCHULTZ
MANAGING EDITOR: CHERYL VAN SISE
EDITOR: M'LOU PINKHAM
EDITOR: NANCY VALENTINE
ASSOCIATE EDITORS: JAMIE MATUSOW, JEAN CASSIDY

WRITERS IN THIS ISSUE
ALEX ANDRIANAPOULOS, BRIAN BARBASH, JIM GREEN, BRET HARTMAN, MARJAN HERICKO, MATJAZ JURIC, SANJEEV KUMAR, ANNE THOMAS MANES, NORBERT MIKULA, JP MORGENTHAL, CAROL MURPHY, BRIAN REED, SEAN RHODY, DIMPLE SADHWAMI, GUNJAN SAMTANI, ALEX WOLOSHIN

PUBLISHER, PRESIDENT, AND CEO: FUAT A. KIRCAALI
VP, PRODUCTION & DESIGN: JIM MORGAN
SENIOR VP, SALES & MARKETING: CARMEN GONZALEZ
VP, SALES & MARKETING: MILES SILVERMAN
VP, EVENTS: CATHY WALTERS
VP, BUSINESS DEVELOPMENT: GRISHA DAVIDA
CHIEF OPERATING OFFICER: MARK HARABEDIAN
CHIEF FINANCIAL OFFICER: BRUCE KANNER
ASSISTANT CONTROLLER: JUDITH CALNAN
ACCOUNTS PAYABLE: JOAN LAROSE
ACCOUNTS RECEIVABLE: JAN BRAIDECH
ACCOUNTING CLERK: BETTY WHITE
ADVERTISING ACCOUNT MANAGERS: MEGAN RING, ROBYN FORMA
ASSOCIATE SALES MANAGERS: CARRIE GEBERT, ALISA CATALANO, KRISTIN KUHNLE, LEAH HITTMANN
CONFERENCE MANAGER: MICHAEL LYNCH
SALES EXECUTIVES, EXHIBITS: MICHAEL PESICK, RICHARD ANDERSON
SHOW ASSISTANT: NIKI PANAGOPOULOS
ART DIRECTOR: ALEX BOTERO
ASSOCIATE ART DIRECTORS: CATHRYN BURAK, LOUIS CUFFARI
ASSISTANT ART DIRECTORS: RICHARD SILVERBERG, AARATHI VENKATARAMAN, TAMI BEATTY
WEBMASTER: ROBERT DIAMOND
WEB DESIGNERS: STEPHEN KILMURRAY, CHRISTOPHER CROCE, BILL DARRON
CONTENT EDITOR: LIN GOETZ
JDJSTORE.COM: ANTHONY D. SPITZER
CUSTOMER SERVICE MANAGER: ANTHONY D. SPITZER

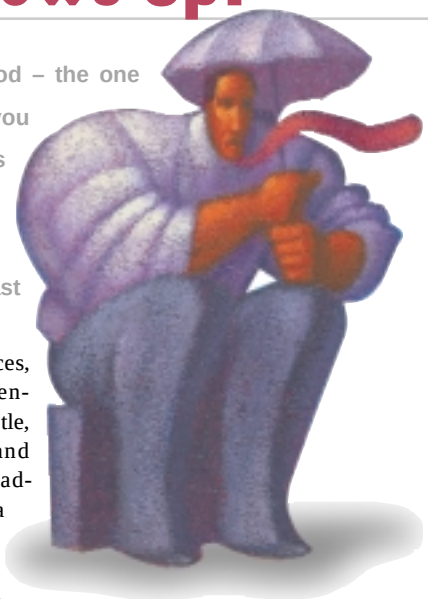
SUBSCRIPTIONS
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT.
SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM
COVER PRICE: $6.99/ISSUE
DOMESTIC: $69.99/YR (12 ISSUES)
CANADA/MEXICO: $99.99/YR OVERSEAS: $129.99/YR
(U.S. BANKS OR MONEY ORDERS)
BACK ISSUES: $10 EA., INTERNATIONAL $15 EA
SUBSCRIBE@SYS-CON.COM

EDITORIAL OFFICES
SYS-CON PUBLICATIONS, INC.
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
IS PUBLISHED MONTHLY (12 TIMES A YEAR) BY
SYS-CON PUBLICATIONS, INC., 135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
PERIODICALS POSTAGE PENDING
POSTMASTER: SEND ADDRESS CHANGES TO:
WEB SERVICES JOURNAL, SYS-CON PUBLICATIONS, INC.
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

Web Services: EAI Grows Up?

Written by
Sean Rhody

Author Bio:
Sean Rhody is the editor-in-chief of ***Web Services Journal****. He is a respected industry expert and a consultant with a leading Internet service company.*
SEAN@SYS-CON.COM



Remember that kid in the neighborhood – the one who had the bat and ball, and if you wanted to play together, you had to use his stuff? Even if he was a pretty nice kid, there was always a time when you longed for the ability to play on your own, or at least use a bat that was more your size.

Until recently, with the advent of Web services, that's been the situation with EAI products. The vendors in the EAI space have been in a constant battle, competing with one another for market share and mindshare in a fairly limited market. And while adding new features to their products has been a strong point for each vendor, interoperability between EAI offerings has been extremely limited. Realistically, it isn't in any vendor's best interests to interoperate – they lose revenue and potentially even an account by doing so. And since their offerings are geared toward integration within a company, the party line is that you need only one EAI product.

Which is fine if your company never merges with another company, or gets acquired, or spun off, or any of the myriad of different corporate dances that can ask IT shops to combine systems they never dreamed of supporting.

An even more problematic issue is dealing with external entities within the supply chain – the suppliers, distributors, and customers. Far and away the largest technical issue with tightly integrated supply chains is the integration of all of the needed systems in the chain. Rarely does an entire supply chain present a neat, unified API for each segment of the business.

But that's just what Web services provides – a way to put that interface on the systems. A common language, allowing two or more participants to work together, regardless of hardware and software.

EAI vendors have been slower to react to the Web services movement than have the application vendors. In part this had been due to Web services' origin in the application server communities. The companies leading the charge on Web services, Microsoft, IBM, BEA, and the like, all have a stake in the way applications are created, and much of the industry perception of Web services is a result of their marketing arms, who would love you to believe that you need to select an application server platform to do Web services.

Adding to the delay for EAI vendors has been the chicken-and-egg problem: vendors were waiting to see if Web services was a flash in the pan before investing limited development resources into it.

But this issue shows that they've begun to adjust and react. And they bring a completely different take on Web services to the party. For EAI vendors, Web services are about integration, not application.

I've long been a proponent of a richer set of Web services protocols – additions to the standard set of UDDI, XML, and WSDL. Because until the technology has a business use, it's nothing but interesting technology. And one of the things that's missing is Business Process Management (BPM). The ability to coordinate activity is crucial to Web services.

Now some people would claim that EAI and BPM are two entirely different things, and if you look at the technologies used to implement them, I won't disagree. But if you look at what they both do, which is to orchestrate communications between entities, then they begin to look very similar. Web services needs EAI capabilities.

At the same time EAI needs Web services. EAI needs to move to an open standard so businesses can begin to take advantage of integration capabilities, and not just within the confines of a single corporation, but also within the whole supply chain. Forget the kid with the ball and bat, we're in the big leagues now.

Security Strategy for EAI

Unify to reduce risk, cost, and complexity

Bret Hartman

Bret Hartman is CTO of Quadrasis, a business unit of Hitachi Computer Products (America), Inc. He has over 20 years' experience in information security and secure systems development. His expertise includes distributed-component security, policy development and management, and security modeling and analysis. Bret is a nationally recognized expert on multiple distributed applications technologies, and an author, regular speaker, and expert panelist on a variety of secure distributed-system topics.
BRET.HARTMAN@QUADRASIS.COM

In today's global economy, organizations are expanding their market opportunities by extending their reach. Mergers and acquisitions, new partnerships, and new business models – including e-business and Web services – are changing the way companies interact with their customers, and with each other. Yet these same initiatives are creating tremendous challenges for the IT groups faced with making it all work.

Today's extended enterprise model is creating complex, distributed IT infrastructures – vast networked environments that comprise hundreds of different systems and dozens of different applications across multiple-partner organizations. To meet this challenge, many organizations are turning to Enterprise Application Integration (EAI) strategies that reduce the cost and time associated with development, integration, implementation, and management of their distributed systems and applications.

This complex, heterogeneous environment doesn't just present new interoperability challenges; it also presents serious privacy and security challenges. No longer is the "back office" hermetically sealed off from the outside world. In exposing critical business functions to suppliers, customers, and employees via the Internet, institutions can expose data, applications, and systems to a variety of potential threats – both internal and external. Meanwhile, users expect that sensitive corporate and personal information will be readily available to those authorized to see it, while securely protected from access by everyone else.

To address security needs, organizations have deployed a variety of point security solutions for each application or system – a situation that increases complexity for both users and administrators. Users of multiple services or applications must remember multiple user IDs and passwords, which is not user friendly and increases security risk. On the operational side, security administrators must manage security policies for each user – for authentication, authorization, and audit – across numerous administrative interfaces. As the number of users, applications, and systems increases, this complexity becomes extremely costly to manage – and increases the chances of a breach through which a hacker or a disgruntled employee can slip in unnoticed.

How can organizations manage this complexity while enhancing security? Just as they have turned to EAI architectures to streamline integration of their distributed applications, they need a comprehensive architecture for Enterprise Application Security Integration (EASI). This framework, which leverages existing security services and applications, enables organizations to meet the critical demand for security across their entire extended enterprise, while reducing risk, cost, and complexity.

A comprehensive EASI framework enables organizations to address a range of critical business and technology requirements, including:

- ***Establishing trust*** with end-to-end accountability across all systems and applications, from perimeter security to mid-tier security to back-office security
- ***Managing complexity*** by providing a single, comprehensive solution for managing security policies across the entire heterogeneous infrastructure of today's extended enterprise
- ***Preserving existing investments*** by leveraging existing best-of-breed security solutions
- ***Accommodating evolution*** through adherence to open technology standards

So what, exactly, is an EASI framework? It's a flexible, standards-based framework that integrates security technologies and products from multiple vendors across the perimeter, middle, and back-office tiers – both within a single enterprise and across multiple enterprise domains. It simplifies the unification of complex security infrastructures by providing the key P's of security integration:

- ***Programming interfaces*** that simplify cross-domain integration today and permit cost-effective future evolution
- ***Policies*** enabling centralized definition and security management across a variety of diverse security products
- ***Protocols*** leveraging open standards, including XML and SAML (Security Assertion Markup Language), an XML-based standard for defining application-independent authentication/authorization credentials
- ***Products*** enabling seamless interoperation of third-party products for authorization, authentication, cryptography, accountability, and administration

The result is a single, virtual "business engine" that unites disparate technologies to address the four A's of enterprise security: *Authentication*, *Authorization*, *Accountability*, and *Administration*. This seamless, distributed framework can enhance end-to-end security, minimize disruption to the existing security infrastructure, and maximize ROI.

From a user's perspective, this means enjoying the simplicity and convenience of Single Sign-on (SSO) when accessing multiple services or applications. From the administrator's viewpoint, EASI enables centralized management of the entire distributed security infrastructure, with end-to-end audit and alerts. For enterprise management, EASI represents a flexible solution for security interoperation that reduces risk while preserving technology investments and accelerating time to deployment.

As organizations continue to extend their reach through innovative e-business models – and as the list of potential threats grows – there is little question that the need for distributed security will increase. By providing a flexible, standards-based integration architecture, an EASI framework can be the key to profitable, new capabilities – while closing the door to information security threats.

written by Brian C. Reed

Data –A Key Part of Web Services

Web services has been touted as the next big achievement that may eventually replace our current model for building e-business applications. Although it's true that Web services presents tremendous opportunities, it is also true that Web services is an extension of traditional environments and tools, and will still require the fundamentals that govern the use of current environments and tools.

AUTHOR BIO:
Brian C. Reed is vice president of business development and strategic planning at DataDirect Technologies, a leading supplier of components for connecting data and applications across diverse environments such as Java, .NET, and legacy, where he leads their business strategy and growth efforts. Brian has been involved with information, databases and connectivity standards such as JDBC, ODBC, ADO, and XML since their inceptions and has more than 10 years of experience in standards-based data connectivity and integration.
BRIAN.REED@DATADIRECT-TECHNOLOGIES.COM

The power of Web services is its reliance on simplicity, a foundation in current Internet technologies, and a focus on standards, including XML for data, SOAP for transport, WSDL for location, and UDDI for directories. Web services applications, like all applications, require a development environment and language with a solid deployment infrastructure, including the ability to connect and integrate existing resources, such as data and processes, as well as develop new ones.

For the purpose of this article, we'll focus on the role of data and XML within Web services as a critical component of the architecture. We'll discuss topics that organizations should consider as they approach data and database utilization with XML and Web services. Assessing the state of the industry in terms of the role of data and XML leads to three interesting topics that we will explore: a database-centric approach, a development platform approach, and the emerging APIs and components.

Database-Centric Approach

Many organizations will approach Web services starting with core assets that already reside within the business – data and processes. For XML Web services and databases, the market has consolidated into two primary categories that could be termed "RDBMS with XML" and "Native XML Database." Leading RDBMS vendors, such as IBM (DB2), Microsoft (SQLServer), and Oracle (Oracle9*i*), have implemented various elements of XML support by extending their products. A few vendors, such as Software AG, eXcelon, and Ipedo, have delivered completely new databases built from the ground up for XML.

From the RDBMS vendor perspective,

TABLE 1: Database Vendor Overview of XML Web Services Capabilities

	IBM DB2	Microsoft SQL Server	Oracle
Primary XML and Web services "packages"	IBM DB2 database, IBM DB2 XML Extender	Microsoft SQLServer database, SQLXML package, SQLServer Web Services Toolkit	Oracle 9*i* database, Oracle XML Developer Kits (XDKs)
Capabilities	Query, Update, Storage	Query, Update, Storage	Query, Update, Storage
XML Data types	XMLVARCHAR XMLCLOB	No specific type	XMLType
XML relational mapping	DAD XML Collections DADX Web services	XML Views XML Templates	XML SQL Utility XSQL Servlet
XML processing	UDFs, Triggers	OpenXML … FOR XML() extension to SQL UpdateGrams	DBMS_XMLGEN SYS_XMLGEN
Stored procedure access	Yes	Yes	Yes
SQL access	Yes	Yes	Yes
Access interfaces	Java extensions JDBC SOAP	HTTP, Xpath, XQuery ADO, ADO.NET SOAP	XPath, Oracle OCI extensions, JDBC SOAP

there are three typical implementation considerations for XML and Web services:

- ***Data types:*** The ability to store XML documents in the database (typically using a data type similar to a LONG VARCHAR or CLOB)
- ***Mapping:*** Access and mapping between hierarchical XML and rectangular data stored in traditional relational database rows and columns
- ***Processing:*** External processes and extensions to stored-procedure grammar and functionality to enable processing of XML.

Let's explore the individual database capabilities with the summary overview provided in Table 1.

IBM DB2

With DB2 and IBM DB2 XML Extender, IBM's approach to XML and Web services includes XML documents in the database with new XMLVARCHAR and XMLCLOB types, XML collections for XML-to-relational data mapping, and a combination of stored procedures, triggers, and user-defined functions (UDFs) for XML processing. DB2 XML Extender leverages document access definition (DAD) files for the schemas that map the hierarchical structure of XML documents to the actual relational structure in the database. In addition, DB2 can be used to create Web services that are defined by SQL statements or by a DB2 XML Extender DAD file enabled through IBM's document access definition extension (DADX). DADX files also manage database connection information using JDBC and JNDI and related XML schema, SOAP, and WSDL information for developing and deploying Web services. DB2 data is accessible through a variety of interfaces including Java, JDBC, and SOAP. www3.ibm.com/software/data/db2/extenders/xmlext/index.html)

Microsoft SQL Server

Microsoft's SQL Server includes the Microsoft XML SQL package and the new SQL Server Web Services Toolkit. These enable XML-to-relational data mapping and direct Web services support from the database, but Microsoft does not provide direct facilities for storing, indexing, and searching native XML documents in the database. The XML Views facility enables developers to work with SQL Server data through XPath and XQuery with XDR XML Schemas and XSD Schemas as if the database were an XML file. The OpenXML object and the new "… for XML" Transact-SQL grammar enable SQL-to-XML processing.

For direct mapping via XML documents to perform updates, inserts, and deletions to a database, Microsoft also includes Updategrams. The SQL Server Web Services Toolkit creates XML Web services from SQL Server stored procedures or server-side XML templates via SOAP. SQL Server data is accessible through a variety of interfaces for building Web services including HTTP via XPath and XQuery, direct SOAP support, and ADO.NET for .NET Framework developers. (See www.microsoft.com/sql/techinfo/xml/default.asp)

Oracle9i

Oracle has implemented support with Oracle9*i* and the Oracle XML Developer Toolkits (XDKs), enabling storage and manipulation of XML documents in databases via the new XMLType, XML to relational mapping, and new PL/SQL procedures for XML processing. PL/SQL package extensions for SQL-to-XML processing include DBMS_XMLGEN (for queries to XML) and SYS_XMLGEN (for query arguments to XML). The XML SQL utility generates Java XML documents, DTDs, and schemas from SQL queries. The XSQL Servlet combines XML, SQL, and XSLT in the server at runtime to deliver dynamic Web content. For Web services, SOAP-based interfaces are provided through Oracle SOAP. Oracle XDKs are available for a variety of development environments, including Java, C/C++, and PL/SQL, with access interfaces including Oracle OCI extensions, HTTP XPath, SOAP, and JDBC. (See http://otn.oracle.com/tech /xml/xdkhome.html)

Keep the fundamentals for your current environments and tools

Native XML Databases

Native XML databases focus on storing and processing XML data and documents natively – storing, reading, writing, and maintaining data and documents in a native XML format. Most XML databases provide their own proprietary tools and interfaces for accessing and manipulating their systems as well as standard interfaces and technologies such as HTTP XPath/XQuery, SOAP, ODBC, JDBC, and more. One advantage of native XML databases is that developers can interoperate exclusively in the XML domain and leverage the latest XML features and optimizations, although these systems may not have all the "burned in" enterprise features of relational databases. Most XML databases include tools for integrating, or migrating, and other types of data such as relational data. Vendors in this category include SoftwareAG, eXcelon, and Ipedo. (See www.softwareag.com/tamino, www.exceloncorp.com/platform/extinfserver.shtml,www.ipedo.com/html/products_xml_dat.html)

Development Platform Approach

While some organizations will focus on XML and Web services using the data-centric database approach, others will focus on XML and Web services using a development platform approach. Because XML and Web services are all about standardization and interoperability, it's important to look to standards-based techniques for data connectivity. From this perspective, the de facto or emerging standards for data connectivity that play a role in XML and Web services include 3GL/4GL environments, Microsoft .NET, and J2EE.

A significant shift is occurring in data connectivity as we move from traditional 3GL/4GL client/server and Web application environments to distributed applications and Web services. Traditional client/server and Web application-environments relied on connection-based synchronous communication with the back-end database. Although this means tight control over locking and transaction behavior, it also can raise some potential performance and scalability issues when you consider the thousands of possible users and transactions that are concurrently updating data across the Web. As a result, Microsoft, with ADO.NET, and the Java community, with WebRowsets, are focusing on XML and improved scalability with XML rowset-type architecture that is fundamentally disconnected and asynchronous in nature. This issue is mentioned here because developers and software vendors must recognize the new paradigm and adjust appropriately.

> Developers choose technologies like Java and XML for portability, but quickly find that the XML-to-database portability goes only so far

3GL/4GL Environments

Although there is tremendous excitement around Java and emerging Microsoft .NET technologies, studies indicate that over half the application development activities around the world still involve traditional 3GL and 4GL environments, from C/C++ to Visual Basic. Given the tremendous investment in technologies and skills, many organizations want to leverage these resources while tapping into the new opportunities that XML and Web services offer.

For data connectivity in these environments, the dominant standard is Open Database Connectivity (ODBC). ODBC is a mature, proven standard for interoperability that is available across all major operating systems and development environments accessing hundreds of different data stores. However, it is not widely known that ODBC includes XML capabilities, and these capabilities can be leveraged to assemble Web services as shown in Figure 1.

ODBC drivers that provide full ANSI SQL-based read-and-write capabilities across XML files are available. Developers can code in the ODBC API they know and manipulate XML using traditional SQL methods and rectangular data formats. They can query XML documents using SQL SELECT, create XML documents using SQL CREATE TABLE, and update XML documents using SQL UPDATE. Developers can also persist the results from an SQL query into an XML document for posting on the Web or to use within a Web services component. These technologies leverage standard XML DTDs and DOM parsers. The code fragment in Listing 1 shows an example of selecting data from an XML file Inventory, XML using SQL and relational methods, and persisting SQL data from Oracle into an XML file called Parts.XML. (See www.datadirect-technologies.com/tech_library/ddlibrary.asp)

> The power of Web services is its reliance on simplicity, a foundation in current Internet technologies, and a focus on standards

J2EE Platform

Standardized data connectivity has been a key part of the Java platform from the early development of Java through the Java Database Connectivity (JDBC) specification. JDBC 3.0 delivers developer productivity enhancements with fine-tuning capabilities, as well as new XML support. With this version, the Java Community is extending the platform with initial work on WebRowsets for XML. WebRowsets are components that exist on top of a JDBC driver to enable the developer to use data in an XML format and a disconnected state (see Figure 2). WebRowsets are in further development under JSR 114 with advancements expected later this year.

Disconnected WebRowsets are rows of tabular data cached in memory that are updatable and serializable to XML. WebRowsets are in memory caches of XML data, currently implemented for tabular or rectangular data, with semi-structured hierarchical data likely to follow. Objects called Writers are available to propagate changes back to the underlying database. Because of the disconnected model, WebRowset uses optimistic concurrency. WebRowsets can be operated upon by standard XML technologies including JDOM. Listing 2 shows an example of using a relational database table inventory as an XML WebRowset.

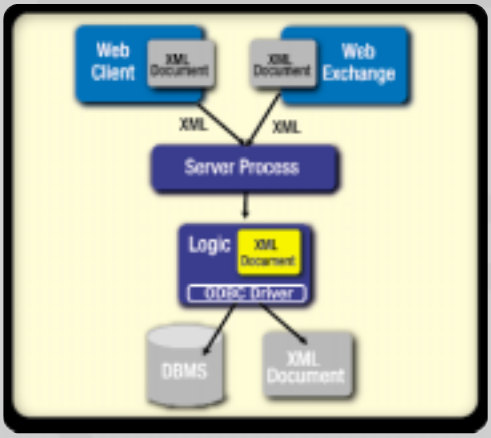


FIGURE 1 **XML example for 3GL/4GL/ODBC**

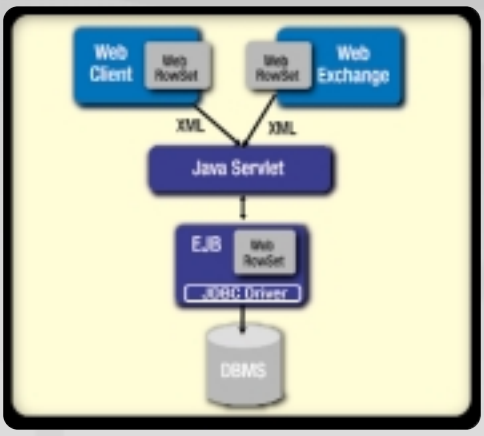


FIGURE 2 **XML example for J2EE-Web Browser**

With WebRowsets, the Java strategy has been to focus on simplicity and portability of data with XML at the expense of some finer control, such as using primary/foreign keys. In the end, for building XML Web services components and applications in Java, developers can leverage JDBC for direct connectivity to a wide variety of back-end data stores through relational structures and SQL grammar, or they can use WebRowsets to share data in XML format under a disconnected model. (See http://developer.java.sun.com/developer, www.data_direct-technologies.com/tech_library/ddlib rary.asp)

Microsoft .NET Platform

With the key focus on XML Web services, Microsoft has fundamentally changed their approach for accessing data to the disconnected model by revamping their development platform with Visual Studio .NET and revised ADO technology to create ADO.NET. From the data access and database perspective for the .NET Framework, Microsoft provides facilities to manipulate data as a tabular rowset or a hier-0archical XML document (see Figure 3).

Microsoft has replaced the connected RecordSet object with a new set of objects led by DataSets. DataSets are essentially in-memory caches of database data in XML format running in a disconnected model using optimistic concurrency. DataSets contain collections of DataTables that are populated from back-end databases accessed via ADO.NET Managed Providers. ADO.NET includes a series of additional objects and methods for manipulating XML data including create, query, and update. Microsoft provides the capability to manipulate the data from an XML approach using XPath, XQuery and so on, as well as tabular data oriented approaches with DataReaders and DataAdapters. Listing 3 shows an example of using a relational database table Inventory as an XML WebRowset.

With ADO.NET, Microsoft's strategy has been to focus on database functionality and a broad programming model, at the expense of database neutrality where key functions, like database metadata and standardized parameter markers, are not exposed. In the end, for building XML web services components and applications with .NET, developers can focus on developing XML applications in a variety of languages using XML meth-ods or database-oriented methods for data manipulation. (See www.microsoft.com/sql/techinfo/xml/adonetprimer.asp,www.datadirect-technologies.com/tech_library/ddlibrary.asp)

Emerging APIs and Components

As we've explored database-centric and API-centric approaches, one thing is clear – there are many ways to use data with XML to build Web services (see Figure 4). This makes you realize the need for a more consistent approach to building applications and Web services that use XML and database data.

In fact, there are emerging technologies that take the combined elements of the other approaches to enable easier interoperability between existing relational data and XML formats regardless of the backend data store (see Table 2).

These software components and tools, for example, use SQL, XML, and environments such as Java to help developers build applications that insulate them from all the variations of database and XML implementations. Interfaces provide traditional SQL grammar extended with key XML constructs to enable the application to seamlessly move between XML and SQL database formats. The opportunity for success lies in their independence of database or platform combined with leveraging other standards including DOM, SAX, JDOM, W3C schemas, DTDs, XPath, XQuery, JDBC, ODBC, and so on. Developers choose technologies like Java and XML for portability, but

TABLE 2: Translating to XML

Sample SQL92 Query	*Resulting XML SQL Query*
SELECT EmployeeID, FirstName, LastName, Title, HireDate, Salary FROM Employees e WHERE HireDate >= {d '2000-01-01'}	xml_document(xml_element('result', SELECT xml_element('Employees_Info', xml_attribute('ID', e.EmployeeID), xml_element('name', xml_element('first', e.FirstName), xml_element('last', e.LastName)), xml_element('title', e.Title), xml_element('hiredate', e.HireDate), xml_element('salary', e.Salary)) FROM Employees e WHERE e.HireDate >= {d '2000-01-01'})
Sample Tabular Data	*Resulting XML Data*
Sample Tabular Data Resulting XML Data 456321|Jane|Smith|Director |2000-01-01|120000 456322|Roger|Jones|Sales Rep|2000-02-01|80000	<?xml version="1.0" encoding="UTF-8" ?> <result> <Employees_Info ID='456321'> <name> <first>Jane</first> <last>Smith</last> </name> <title>Director</title> <hiredate>2000-01-01</hiredate> <salary>120000</salary> </Employees_Info> <Employees_Info ID='456322'> <name> <first>Roger</first> <last>Jones</last> </name> <title>Sales Rep</title> <hiredate>2000-02-01</hiredate> <salary>80000</salary> </Employees_Info> </result>

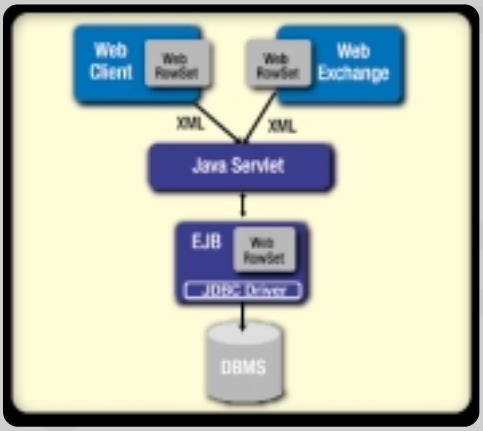


FIGURE 3 | XML example for .NET–ADO.NET

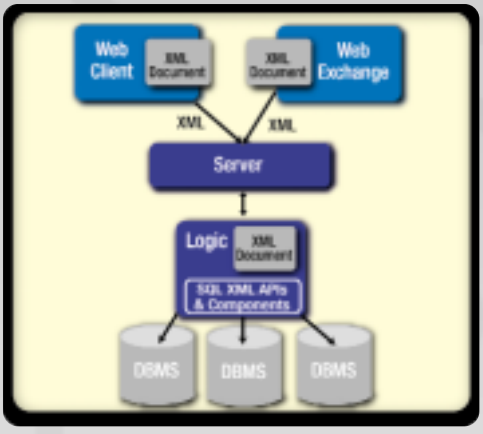


FIGURE 4 | Emerging APIs and Technology

quickly find that the XML-to-database portability goes only so far. Vendors focusing on emerging technology here include RogueWave Software and DataDirect Technologies. (See www.datadirect-technologies.com/jdbc/connectjdbc/jpxconnectjd zbc.asp.)

Conclusion

Although some organization may have a utopian view – where all Web services data resides in XML – the reality is that trillions of bytes of corporate data are stored in existing relational and legacy databases and will continue to reside there. XML will likely become the common integrating medium for Web services applications, providing a de facto standard to transmit corporate data. Ultimately, data will typically start and end in a relational or legacy database. As we've seen, developers have a variety of interfaces and tools at their disposal to interoperate between database data and XML data and build new types of applications and Web services. The industry will evolve and consolidate around a handful of them.

Listing 1:

```
// Sample SQL SELECT of XML data
// Inventory.XML is the XML file containing
// well-formed hierarchical XML data
SQLDriverConnect (hdbc, DSN="XMLDataSource"…)
SQLExecDirect (hstmt, "Select * from Inventory",
SQL_NTS)

// Sample SQL SELECT with relational to XML persistence
// Parts is the relational table of parts information
// that will be persisted into XML file
SQLDriverConnect (hdbc, DSN="OracleDataSource"…)
SQLExecDirect (hstmt, "Select * from Parts", SQL_NTS)
SQLSetStmtAttr  (hstmt, SQL_PERSIST_AS_XML,
"//myweb/inventorylist/parts.xml", SQL_NTS)
```

Listing 2:

```
// Connect and execute statement
stmt = conn. createStatement();
rs = stmt. executeQuery(" select * from inventory");

// Create and populate the WebRowSet
WebRowSet wrs = new WebRowSet();
wrs. populate(rs);

// close the connection
conn. close();

// Iterate through WebRowSet that contains values...
wrs. next();

// write the RowSet out as XML
wrs. writeXML( out);
```

Listing 3:

```
//Create statement
SqlCommand command = new SqlCommand();
command.CommandType = System.Data.CommandType.Text;
command.CommandText = "select * from inventory";
command.Connection = con;

//Populate DataSet with results of select
DataSet dsInventory = new DataSet();
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = command;
adapter.Fill(dsInventory,"Inventory");

//Get DataTable from DataSet
DataTable myDt = dsInventory.Tables[0];

// Iterate for each row
foreach (DataRow r in myDt.Rows)
  ...

//Write DataSet as XML
dsInventory.WriteXml(writer);
```

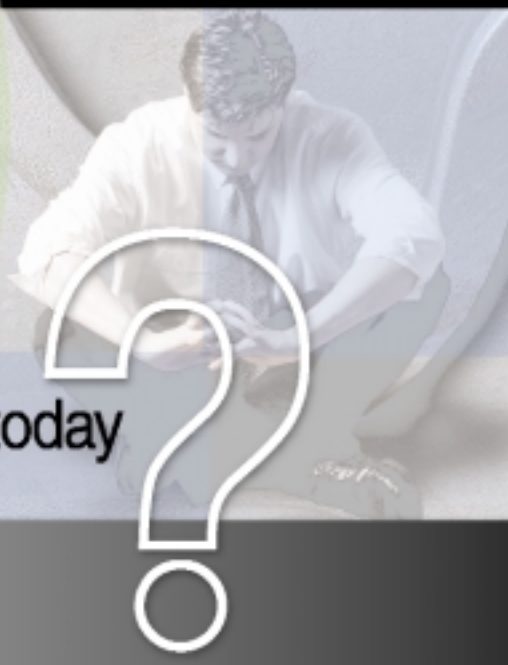
Download the code at sys-con.com/webservices

FOCUS ON EAI

written by JP Morgenthal

Web Services for Enterprise Application Integration

What solutions are offered for the enterprise today



Enterprise applications have really made significant strides over the past 10 years (especially in the past 4) to improve their ability to integrate into a larger corporate scheme. There was a time when the letters SAP invoked uncertainty on the part of non-SAP consultants as to what these systems did. Back then, most integration was done primarily through the underlying data model and information on the platform was scarce, especially on the SAP Web site.

JP Morgenthal is CTO for Ikimbo and an internationally prominent expert on the design and implementation of distributed systems for the enterprise, and underlying technologies: Java, XML, enterprise application integration (EAI), business-to-business (B2B), and supply-chain management (SCM). JPM@IKIMBO.COM.

The integration with SAP that wasn't handled through customization of the data model occurred mainly through extracted documents known as IDOCs – structured documents based on internally defined structure and loosely defined codes. This made IDOCs useful for integration internally among developers within the same organization. As things progressed, SAP customers demanded a better interface for real-time integration of Enterprise Resource Planning (ERP) capabilities with existing manufacturing systems. Thus, Business Application Programming Interfaces (BAPI), offering an Object Request Broker (ORB) architecture against which programmers could communicate with ERP objects was born. With BAPI, it became possible to develop applications that could synchronize the data in the ERP system with other existing enterprise applications, such as Electronic Data Interchange (EDI).

BAPIs worked well for a time, until the ERP information had to be shared with external entities, such as the supply chain. Many companies didn't like the idea of allowing their trading partners and customers to directly impact their mission-critical systems, and IDOCs offered little assistance in this area, given the lack of structural information that accompanied the documents. Since this need emerged in parallel with the development of the eXtensible Markup Language (XML), the requirement and specification were in complete synergy to become the natural solution.

With the advent of Web services – a model for developing application components designed for sharing with other applications in a loosely coupled manner—the question to be answered is, "What problem does this solve for enterprises today?" As we've seen thus far, each of the advances in technology was incorporated into the enterprise application platform based on consumer needs. Initially, the need was customization, then communication of data with other internal systems, and finally, the need for real-time access for synchronization.

Web Services in An Enterprise World

Business-to-business (B2B) and enterprise application integration (EAI) are both about the development of new or modified processes for the purposes of bettering the organization. These enhancements might lead to faster production, lower cost of development, or increased communication with customers and partners. The crux of these enhancements can be readily seen in Customer Relationship Management (CRM), ERP, Supplier Relationship Management (SRM), and Supply-Chain Management (SCM) applications.

B2B and EAI are also about providing access to data that has been typically developed in stove-pipe applications locked away from each other by artificial boundaries. These new processes require that information be provided on demand and be up-to-date with the latest changes. It's no longer acceptable to have a two-day lag on inventory information in a shop that relies on Just-In-Time manufacturing. Large

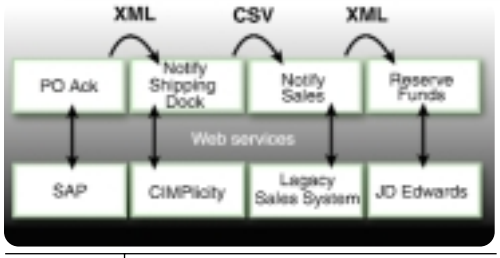


FIGURE 1 | **SOAP encoding of XML documents**

manufacturers, such as General Motors, rely on completed parts assemblies arriving at the shipping dock the day they are to be assembled into a vehicle.

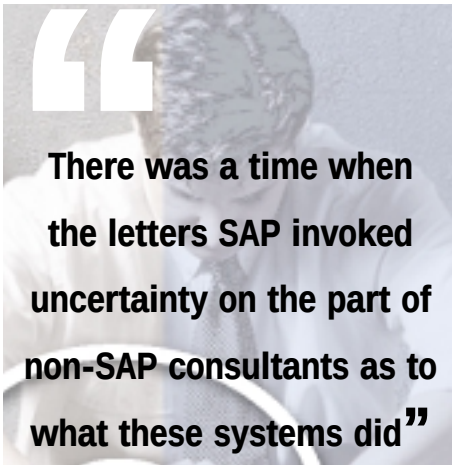
To answer the question posed at the end of the last section, we need to look at the Web services offering from different perspectives. The first perspective is the traditional view of Web services that many programmers are being introduced to today through tools such as BEA's WebLogic Workshop and Microsoft's Visual Studio .NET. This perspective leverages the Simple Object Access Protocol (SOAP) and Web Service Definition Language (WSDL) as a means to wrap a functional interface with an XML request-and-response document. From this perspective, the functions being wrapped have well-defined method signatures and use known data typing, such as strings and integers. The XML documents encode the parameters and return values to these functions using SOAP encoding (see Figure 1).

This methodology extends the Remote Procedure Call (RPC) functionality provided by popular ORBs, such as CORBA, DCOM, and Java RMI. However, it extends the RPC capabilities to operate over more traditional, and often firewall-friendly, interfaces such as SMTP and HTTP. On the other hand, merely extending an existing methodology does little to demonstrate the role that Web services could play in an EAI or B2B solution. For this we need to look at Web services from another perspective.

In the introductory paragraphs we discussed how systems like SAP used a document metaphor for data exchange with other systems—the IDOC. The weakness in the IDOC approach was the lack of communication of the structure to systems that weren't privy to the underlying data model. This is where the real opportunity for Web services exists.

When we reviewed the traditional approach to Web services, we saw that they operate through the use of XML-based request-and-response documents. XML was chosen as the format for SOAP encoding because it was able to incorporate metadata about the data encoded as well as carry the data itself. But SOAP is also a general enveloping facility that can carry non-XML documents between applications and can use alternate encoding schemes for the data it is carrying. Therefore, it's possible to use the multitude of existing document formats as input to and output from Web services processing.

Viewed from this perspective, it would be possible to use existing spreadsheet models and IDOC-like documents as input into a server-side process, thus providing significant reuse of existing systems architectures for interoperability and integration. Moreover, the SOAP architecture provides ample support to notify the sender that the document type is understood and has been properly processed by the receiver.



You might ask, "Why go through all the trouble of implementing a new encoding scheme instead of just transforming the data into XML?" That's an excellent question, which can only be answered by pointing to a possible real-world scenario. Before XML existed, there was a multitude of data formats, such as comma-separated values, tab-delimited, and even column-specific formats. To facilitate integration with existing applications, these formats were generated by many legacy systems; systems that today are costly to pull out of production or to rewrite to support XML.

Developing a new B2B or EAI process to communicate with systems might require that the data be formatted for the time being in something other than XML. Web services enables us to take advantage of these systems as components of a much larger process without having to first rewrite them – a very powerful and often overlooked capability of Web services. Remember, the goal of Web services is to componentize our legacy systems so they can participate in newly designed processes, which means that they will take some input and provide some output that will be used in downstream steps.

Next Steps for Web Services EAI

In contrast to past supporting changes made to enterprise applications for the purposes of integration, Web services is clearly being driven by the hype surrounding XML and the support for Web services by leading companies, such as Microsoft, IBM, BEA, Oracle, and others. The widespread support for Web services by these companies – for example, Microsoft betting the company on a Web services strategy – has led vendors such as Siebel, SAP, JD Edwards, and others to develop Web services or SOAP interfaces to their products. It's still questionable how soon enterprises will adopt these interfaces for use in new application integration solutions.

Standardization will be a key contributor to this adoption, if and when it occurs. In addition to SOAP, organizations created to ensure the interoperability of Web services implementations (WS-I) and emerging standards, such as Web Service Flow Language (WSFL), will assist with tying together these components in a consistent manner. These will help increase the chances of success.

Note: while there are many contrasts between efforts, the Object Management Group (OMG) has followed a very similar history in the CORBA arena, with only a modest level of adoption. And, while there are many differences between CORBA and Web services that don't allow for prediction based on past experience – for example widespread support by Microsoft – there's certainly a large degree of consistency in the approaches that indicates that Web services will have a much steeper learning curve than most of the early XML-based standards.

FOCUS ON EAI

written by Jim Green

POWERING
Web Services Through Integration Technology

Agreement on standards will lead to a new era on computing

Markets are created when something is provided that didn't exist before. Markets explode when that capability becomes compelling – e.g., the new offering becomes dependable and usable by a large volume of people who derive significant value from it. Web services does the latter. It will cause business use of the Internet, both external and internal to companies, to explode.

Jim Green is CTO and executive vice president of product development for webMethods, a leading provider of integration technology. He has more than 20 years' experience in developing products and technology for distributed computing environments. Prior to joining webMethods, he served as chairman and CEO of Active Software, which webMethods acquired in August of 2000.

Assessing the Hype

Why is this the case? What significant value will organizations derive from Web services? Most of the answers to these question can be found by examining the current state of the integration market. The use of networks to exchange information between applications and companies is referred to as "integration." It's a market, and unfortunately, it's complicated. The data is in different formats, in different places, accessible through complex interfaces, and exchanged by transferring data across unsecure networks.

How do Web services address this issue? In technology, whenever complex problems are addressed, the ultimate solution shows itself by factoring the problem correctly and partitioning the problem into smaller units. Then the total solution is realized by leveraging and using the smaller solutions. A critical factor is that "lower-level" technical issues are hidden from the problem solver at the next level up. Web services has the potential to do precisely this for integration.

A classic example is the personal computer. A PC is composed of a chassis, a motherboard, and various components such as graphics cards that connect into the overall system. Those parts are, in turn, composed of other, simpler, "off the shelf" components such as integrated circuit chips. What are the key components to this success? The appropriate partitioning of the problem, opaque interfaces (e.g., standard and common interfaces that hide the complexity behind them), and standardization, so that problem solvers can easily understand and leverage work done by others.

So what does this have to do with Web services? Let's examine the overlooked aspect of the word "service." In the example of the personal computer, an integrated circuit, generally on a chip, provides a service to the board designer – who no longer has to be concerned with the inner workings or

> **"As we encapsulate the service logic using Web services, more leverage is achieved through abstractions, productivity soars, and the market explodes"**

nonstandard interfaces of the integrated circuits – they just plug in the chips. Similarly, a Web service provides a business function to the service user, who no longer has to deal with the technical APIs of some proprietary application or some obscure communication protocol.

A key element in the hiding of the complexity associated with a particular business is a well-known mechanism for describing the nature of the business function. The Web Services Description Language (WDSL) standard is the means by which Web services hides the complexity of a particular business function. It also allows multiple services to express themselves differently but in a way that can be interpreted by anyone who understands WSDL. Thus, all services are expressed in a common manner, and in a way that hides the complexity behind them (called opaqueness). The normalization of the interfaces is the key to scaling into larger systems without creating expanding complexity.

Today, integration software vendors provide tools to leverage the individual and proprietary APIs of packaged and custom-built applications. This is similar to having a PC manufacturer build a different interface for each graphics card on the market. Even worse is the application-server approach, which is equivalent to having the PC vendors build their own graphics cards out of the lowest-level components, which would require all software to be custom developed for the particular environment on which they are to run. As we encapsulate the service logic using Web services, more leverage is achieved through abstractions, productivity soars, and the market explodes.

Properly designed to expose business functions, these services can be accessed from a heterogeneous application, another business unit, or even across the Internet – all without any understanding of the implementation of the service. That, in short, is revolutionary.

Web services hasn't been overhyped. It's been mis-hyped. It's about integration. Time will show that its promise for changing the way companies do business has actually been underhyped.

What Corporations Want

Today's CxOs have a burning need to more efficiently integrate the numerous proprietary applications and legacy systems that make up their enterprises. Web services applied at the enterprise level can directly address that need.

Over the past several years, companies have invested heavily in their IT infrastructures, particularly in new ERP, CRM, and supply chain management systems that streamline and automate business processes. In the early stages, plugging those systems into existing mainframe and legacy applications (or even making them work with each other) was a very costly and messy business. Making things work together required a lot of hand coding, sometimes with a little help from middleware. This generated the usual problems with flexibility and maintenance, not to mention cost.

Of course, this challenge created an opportunity. Enterprise application integration (EAI) stepped in to bring a "product" approach to the problem. The result was that over half the time and cost involved in integrating systems was removed. And with an integration infrastructure in place, changes could be made to the business processes without having to call in the systems integrator troops.

This has created a new way of looking at existing IT resources. In the past, mainframe applications and legacy databases have been viewed as resources to be connected into. Now, with an integration infrastructure on top – built using the new Web services standard – those resources can be turned into services.

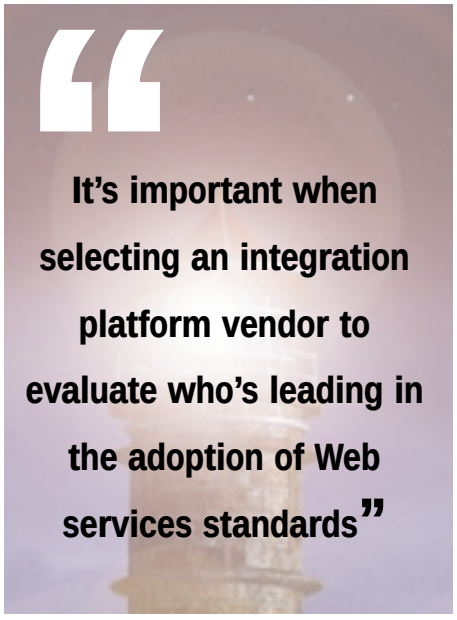
Note that this discussion has centered around leveraging existing applications, because that's where the market is. There are already components and interfaces at the application object development level using the J2EE or .NET technologies. The benefit of Web services is not realized at the local method invocation level between software components. That problem has been addressed. It's at the business-service level (using WSDL) where services are called across the network (using SOAP – Simple Object Access Protocol).

When built using an enterprise integration platform, Web services can be created without writing code. This capability is markedly different than using application servers, which were created to facilitate programming through the use of a single machine language–based object paradigm. The only users of application servers are programmers. The targeted users of an integration platform are nonprogrammers.

In summary, CxOs can leverage their significant costs in their primary mission-critical applications without changing them, without risking their operations, and without writing code.

Enterprise Web Services

Let's state the obvious. You can take any piece of software and turn it into a Web service by simply putting a Web interface on it—that's what WSDL is all about: a complete separation of implementation and interface. There's nothing about a Web service



that dictates its weight or robustness. It could be something trivial built on five lines of code.

Enterprise Web services, however, the kinds of functions serious companies will be using in serious ways, will be used in mission-critical settings and incorporate important attributes such as transactional control, security, management, and the ability to be orchestrated in a business process.

Before companies begin leveraging their internal mission–critical IT investments and/or begin publishing them to the outside world, it will be critical for Web services technologies to meet three fundamental requirements:

1. They’ll have to provide a mature, comprehensive, enterprise mechanism for integration.
2. They’ll have to facilitate the utilization of Web services in conjunction with all other IS resources to define end-to-end business processes.
3. They’ll need to support a management infrastructure that ensures Web service providers are able to meet their service-level obligations.

In particular, the ability to manage an external Web service at the same level or an even greater level than they might internally is critical. It’s the only way companies will allow themselves to depend in a mission-critical fashion on Web service technology.

No Replacement for Integration

In their zeal, some promoters of Web services technologies have advocated that the new standards will replace the need for integration platforms. But let’s be clear: Web services is not synonymous with integration.

Web services as a technology does not cover important integration topics such as data transformation, transactional integrity, queuing systems, publish and subscribe models, human intervention (workflow), adapters to existing interfaces, security concerns, and business process modeling. A full integration system transcends Web services and delivers a much broader set of capabilities.

The Standard Approach

The importance of Web standards in bringing about this evolution cannot be overstated. Standards are more than something to be adhered to – they’re what it’s all about.

The key driver behind what is taking place is the customer’s desire to automate business processes across systems, both internally and externally, and to drive down the cost and time involved. As with all evolutions, it starts with the implementation of basic functionality and then evolves into a set of standards. Standards then lead to open systems, which benefit the customer economically.

The importance of Web services is not that they are cool, or that they are an advancement in technology, but that they represent the beginning of standardization applied to the integration market.

This is huge. It will drive difficulty down, resulting in reduced costs and time. There are ongoing multiple standardization efforts in the industry to facilitate the deployment of Web services. However, a few key specifications have established themselves as the de facto standards associated with Web services. These specifications and technologies address the following core functional areas:

- ***eXtensible Markup Language (XML):*** A key standard for providing a platform-neutral syntax to represent data. Many Web services use XML for representing data and XML Schemas to describe data types. Because of this, Web services are often referred to as XML Web services.
- ***Simple Object Access Protocol (SOAP):*** Defines a lightweight protocol for XML document exchange, and provides a convention for Remote Procedure Call (RPC) using XML messages. It serves as a simple model to envelope and encode request and response messages exchanged between Web services.
- ***Web Services Description Language (WSDL):*** An XML-based contract language that provides a simple way to describe how to invoke a Web service and provide information on the data being exchanged. It can be thought of as an XML-based language to describe Web service interfaces and the protocol bindings for accessing the services.
- ***Universal Description, Discovery, and Integration (UDDI):*** A cross-industry specification for a registry of Web services. It provides a way to register Web services descriptions in a registry, enabling discovery by potential users of the services. Web Services Inspection Language (WSIL) is another specification that assists in the inspection of a site for available services.
- ***Open Management Interface (OMI):*** Another key standard that doesn’t receive much attention. Coauthored by webMethods and HP, OMI is the lingua franca combining integration software with systems and enterprise management solutions – and thus as a standard for managing Web services.

Together, these standards offer a way for applications to be ready-made to work with one another. The result is that things will “snap” together as opposed to having to be “glued” together. That’s a tremendous benefit to corporations because it will allow them to accelerate their integration efforts and dramatically drive down costs.

It’s also a major reason why it’s important when selecting an integration platform vendor to evaluate who’s leading in the adoption of Web services standards. Not because standards are inherently good, but because they will revolutionize the economics of integration.

Where Is It Going?

Where will Web services make its first big impact in business? I firmly believe companies will begin by experimenting with Web services internally. The new Web services standards that have been built into integration platforms will first be used to link internal applications. IT staff will then begin creating new Web services for their internal constituents based on that integration infrastructure. This will happen not only because it offers a low-risk introduction into Web services tech-

nology, but because it also offers the biggest return on investment.

Once the integration groundwork or plumbing is laid, companies will then reach out to their most important customers and trading partners to automate business practices. This viewpoint is shared by Forrester Research, which recently predicted that most Web services will be concentrated around transactional connections internally, and with key trading partners later.

Why will this happen on top of an integration platform? Because it's the only cost-effective way to do it. Just think of the challenges involved in creating a Web service that leverages customer information. In most typical Global 2000 companies, customer information is stored in a wide range of systems, from mainframe and legacy systems and databases, to newer CRM and ERP platforms. If you were to build a Web service that required access to customer information, you might have to go back in and create connectors to 10 different back-end systems to get all of the information that makes up the whole customer picture.

Corporations want to be able to gain access to and create that whole customer record for their own purposes now, without even planning ahead to the day when they will roll out Web services. That's why so many are moving ahead with internal integration programs. But once that integration is in place, it becomes a relatively simple matter to create a Web service to allow others to leverage that customer information in new ways.

With the new technologies, it can be done without disrupting the existing business rules and business processes, by adding a service interface to the application(s) and exposing that to service consumers. It's simply another form of integration.

From the consumer's viewpoint, if you're going to orchestrate a Web service into your business process, you will require that Web service to be reliable, to protect against potential harm to your business. Subscribers of Web services, or any computer service for that matter, have generally looked to define a service level agreement, something that binds the provider to meet certain service levels at the risk of paying serious penalties.

That's a fine concept, but you have to be able to manage the Web service first. As Forrester notes, the industry is still far from agreement on Web services management. Management and monitoring (as well as security) are issues that an enterprise-scale integration platform addresses today, and why Web services will not replace the need for integration.

The Complete Picture

By reaching a consensus around new Web services standards, the computing industry is about to usher in an exciting new era.

The standards are already helping to drive down the costs and time involved in integrating systems, and that will accelerate the adoption of integration technology within corporations. Companies will begin by getting their own houses in order, linking internal applications and databases to create a true corporate-wide picture of their customers and business operations. They will then be able to leverage that infrastructure to roll out enterprise Web services, aimed first at internal constituents, and secondly at their most important customers and business partners.

As new standards are brought to bear addressing key issues such as security, support for transactions, and business process management, we will begin to see the emergence of a wide range of serious public Web services. Benefits will be derived by consumers, platform providers, software vendors, and a new class of service providers. The economics will drive the momentum, and everyone will win.

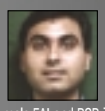
FOCUS ON EAI

written by Gunjan Samtani & Dimple Sadhwani

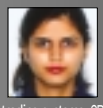
J2EE-Based Application Servers, Web Services and EAI

Support for building, deploying, managing, assembling, and executing Web services

Today, Web services are being portrayed as the building blocks for the EAI platform, whereas, in the last three-to-four years, J2EE-based application servers have been able to carve their way to the core of enterprise application integration (EAI) solutions for several small, mid-, and large-size companies. This article examines how J2EE-based application servers support Web services and how they tie into an enterprise's overall integration and Web services strategy, enabling companies to use service-oriented architecture for EAI.



Gunjan Samtani, divisional vice president of information technology at UBS PaineWebber, has several years of experience in the management, design, architecture, and implementation of large-scale EAI and B2B integration projects. He's the primary author of *B2B Integration: A Practical Guide to Collaborative E-Commerce* www.worldscientific.com/books/economics/p263.html GUNJAN_SAMTANI@YAHOO.COM



Dimple Sadhwani, an independent consultant in the field of e-commerce technologies, has many years of experience working for financial and telecommunication companies on large scale trading systems, CRM applications, Internet/intranet portals, and client/server applications. As well as coauthoring *B2B Integration: A Practical Guide to Collaborative E-Commerce*, she has also authored several articles in the field of Web services. DIMPLE_SADHWANI@YAHOO.COM.

THE MYTH...

Service-oriented architecture–based Web services is a "total and complete" EAI solution by itself–i.e., a solution for all integration needs.

THE FACT...

Web services is not EAI in and of itself. Rather, Web services is just another technology that enables EAI, and it can significantly change the traditional point-to-point integration approach.

Enterprise Application Integration Platform

Enterprise application integration is the process of creating an integrated infrastructure for linking disparate systems, custom and packaged applications, and data sources across the corporate enterprise. As shown in Figure 1, the EAI platform enables this process.

This platform provides a full duplex, bidirectional solution to seamlessly share and exchange data between ERP, CRM, and SCM; databases; data warehouses; legacy systems; and other important internal systems within a company.

J2EE Platform

The Java 2 Enterprise Edition (J2EE) platform provides a complete framework for the design, development, assembly, and deployment of Java applications built on multitiered distributed application models. These applications can reuse components, where a component is defined as a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components.

The J2EE platform has become a proven EAI platform because it provides features such as reliability, scalability, flexibility, platform support, code portability and reuse across tiers, transaction management, security, clustering, load-balancing and fail-over capabilities, performance, availability, connectivity, and Enterprise JavaBeans (EJB) strength.

What Do J2EE-based Application Servers Offer?

J2EE-based application servers are built on the J2EE-standard specifications for services and capabilities. They are component-based products and services that reside in the middle-tier of a server-centric architecture and provide mid-

FIGURE 1 | **The EAI platform**

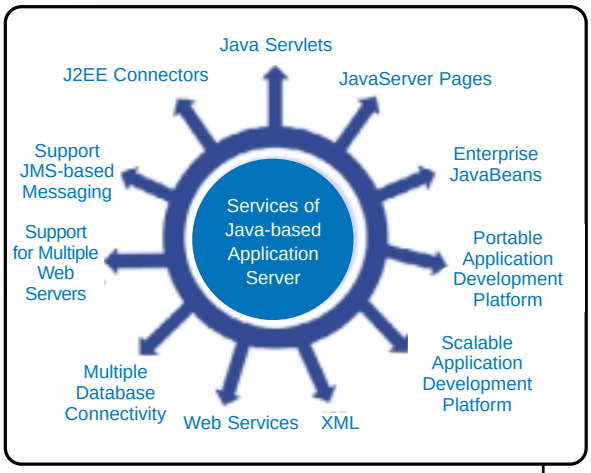


FIGURE 2 **Service of Java-based application server**

dleware services for session and state management, unified security, database connectivity, and persistence (see Figure 2):

- Java servlets
- JavaServer Pages
- Enterprise JavaBeans
  - Portable application development platform
  - Scalable application development platform
  - Extensible Markup Language (XML) Support
  - Multiple database connectivity
  - Support for multiple Web servers
  - Support for JMS-based Messaging
  - J2EE Connectors

Figure 3 depicts where these services fit into a multitiered application, where they are grouped under presentation logic, business logic, and application services.

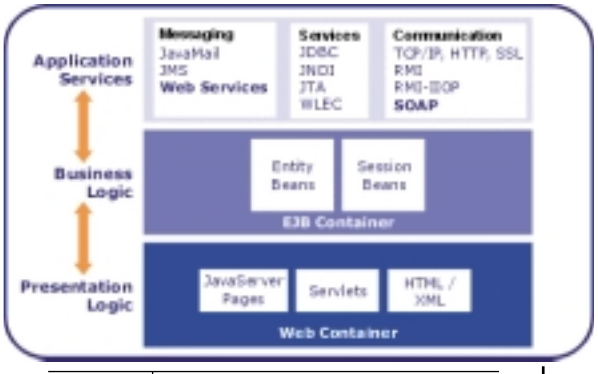


FIGURE 3 **The different tiers supported by J2EE-based application server**

Web Services

Web services, built on service-oriented architecture (SOA), is a stack of emerging standards that describe service-oriented and component-based application architecture. It provides a distributed computing technology for revealing the business services of applications on the Internet or intranet using open and standards-based XML protocols and formats. The use of standard XML protocols makes Web services platform language- and vendor-independent and an ideal candidate for use in EAI solutions. It has the potential to quickly replace older technologies such as CORBA, DCOM, and EDI as the preferred way to tackle the challenges of EAI.

Web Services Framework

A Web services framework for EAI has to provide an integrated development environment and platform for easily building and deploying Web services and service components. There are a few essential features that Web services solution vendors will need to incorporate to successfully support Web services (see Figure 4):

- Easy and secured connectivity to private and public UDDI or any other repository
- Effective audit mechanism through which the access and usage of Web services can be closely monitored
- Efficient security safeguards such as policy management and authentication, for the access and usage of Web services
- Easy development, deployment, publishing, finding, and dynamic binding for Web services interfaces
- A stable environment for rapid development of Web services–based applications

J2EE and Web Services

There's a very strong convergence between J2EE and Web services. The new APIs released by Sun, as part of J2EE 1.3, provide a top-to-bottom, end-to-end solution for a Web services–based architecture. J2EE 1.3 simplifies integration with new technologies for Web services, such as Java Message Service (JMS); J2EE Connector Architecture (JCA), Java API for XML Messaging (JAXM), Java API for XML Processing (JAXP), Java API for XML Registries (JAXR), and Java API for XML-based RPC (JAX-RPC).

J2EE-based application servers have to provide all or a subset of the features of the Web services solution discussed in the previous section. Apart from that, they should allow:

- Invocation of Web services' existing J2EE applications, such as JSP, servlets, and EJBs using SOAP-based messages
- Create Web services out of a stateless Java class, a stateful Java class, and a stateless session Enterprise Java Bean (EJB)
- Wrap existing EJB methods as XML operations
- Convert an existing J2EE application into a service-oriented architecture using Web services
- Enable Web services to inherit all the runtime and life-cycle management elements of J2EE applications
- Provide aJava API for UDDI browsing, querying and publishing a Web service as WSDL

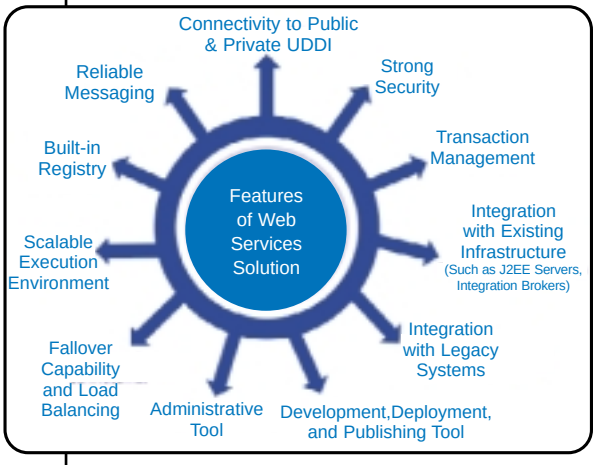


FIGURE 4 **Features of a Web services solution**

It's worth mentioning that the quality of service offered by a J2EE-based application server platform will rely and depend as much on Web services standards and protocols as it will on the maturity, scalability and integrity of the application server itself.

An Example of J2EE and Web Services

The sales department employees of a company use a portal application, running within a J2EE-based application server, to access sales data. As part of the login process, the portal

application has to check the authentication (userID/password combination) and authorization (entitlements) of the users. The information about Web services offered by the entitlements application (which may be published by some other group within the company) is obtained from the private/internal UDDI registry and invoked over the intranet. The implementation of the business methods exposed by the Web service is provided by EJBs contained in another application server. This is a typical example of server-to-server application integration using Web services. The sequence of steps follows:

1. The user enters userID/password information on a JSP/HTML front end that is passed over to the portal servlet running within a J2EE-based application server for validation.
2. The servlet supporting the portal framework gets information about Web services that's made available by the entitlements application by doing a lookup in the private UDDI registry.
3. The location and WSDL binding information of Web services is sent to the servlet as a SOAP-based message.
4. The servlet invokes the Web service published by the entitlements application, passing userID and password as a part of a SOAP-based message.
5. The actual implementation of the Web service is provided by EJBs running within another J2EE-based application server. The EJBs use JDBC API to get information from the data source.
6. The EJBs send the Web services response to the portal servlet as a SOAP-based message.
7. Based on the response, the user is either allowed or denied access to the portal application.

As seen in the example, with the use of Web services, integrated applications no longer have to know the specifics of static information, such as how, when, and where. This completely changes the traditional paradigm of point-to-point integration for EAI.

Leading J2EE Application Servers

Within companies there will be a much wider and faster adoption of Web services as integration components of J2EE applications for EAI, if the current leading application servers, such as BEA's WebLogic and IBM's WebSphere, start providing full and complete support for Web services standards.

Let's quickly take into account the state of Web services support by two of the most popular J2EE application servers – Oracle's9*i* AS, BEA WebLogic, and IBM WebSphere. This information was current as of this writing and based on publicly available information from the vendors.

- ***Oracle9i/AS:*** Oracle9*i* JDeveloper facilitates efficient development and deployment of Web services onto Oracle9*i*AS. OC4J is the runtime platform for Web services. Oracle Enterprise Manager is the Web services management console. Oracle9*i*AS UDDI registry provides the facility to publish, query, and search a Web service.
- ***BEA WebLogic 6.1:*** WebLogic Server hosts and exposes simple one-call, one-response Web services. It offers support for the basic protocols and standards, such as SOAP, with support planned for emerging standards such as UDDI and WSDL. It includes support for XML parsing, transformation, and standard support for JAXP 1.1, SAX V2.0, DOM Level 2, and W3C Schema.
- ***IBM WebSphere 4.0:*** WebSphere Application Server version 4.0 contains integrated support for the key Web services open standards such as Apache SOAP version 2.2, a Java-based implementation of the SOAP 1.1 specification with support for SOAP with attachments; Web Services Description Language version 1.1 processor and generator; Universal Discovery, Description and Integration version 1.0 interface; and UDDI4J, an open-source Java implementation of the UDDI client-side API.
  - ***Apache SOAP version 2.2:*** A Java-based implementation of the SOAP 1.1 specification with support for SOAP with attachments
  - ***Web Services Description Language:*** Version 1.1 processor and generator;
  - ***Universal Discovery, Description and Integration:*** Version 1.0 interface
  - ***UDDI4J:*** An open-source Java implementation of the UDDI client-side API. UDDI4J contains programming interfaces that applications need to publish, find, and bind to a Web service. It also includes the source code and the complete Javadoc for the APIs

Build On Top of Existing Assets

Companies will be able to lower their investment cost in implementing Web services solutions by building them on top of their existing assets. If a company has invested millions of dollars to put their J2EE-based EAI infrastructure in place, it makes absolutely no sense to abandon it and just embrace Web services.

J2EE-based application servers would provide support for building, deploying, managing, assembling, and executing Web service components. Using existing J2EE application servers, companies should be able to use Web services to quickly and easily integrate their J2EE applications with other technologies and applications across their enterprise. This is the only way to achieve a higher return on investment (ROI) in implementing Web services.

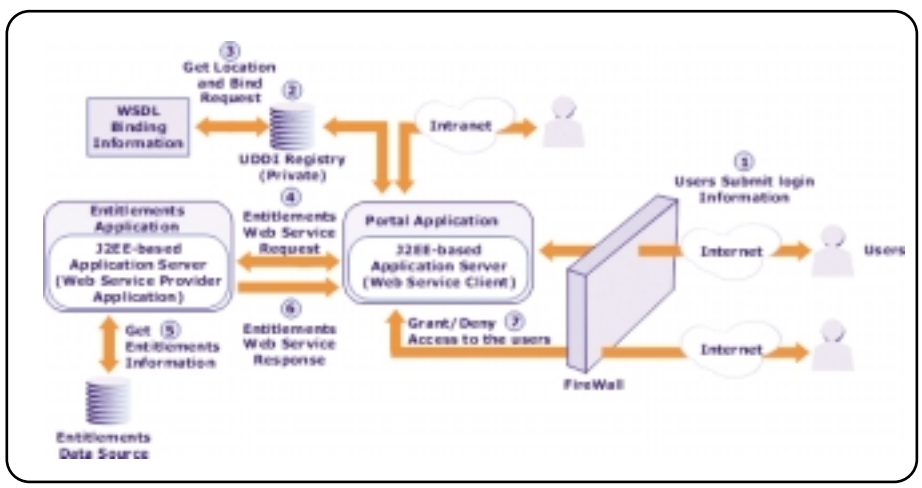


FIGURE 5 | **Example of J2EE and Web services for EAI**

FOCUS ON EAI

written by Alex Andrianopoulos

EAI

The Framework Behind Web Services Integration

An evolution toward lower costs

Standardizing connections between systems is critical for efficiency. We've all heard of Moore's Law stating that processing power doubles every 18 months, but you may not have heard of Gilder's Law that network bandwidth doubles every 6 months. This leaves us with an environment in which connectivity gains grow three times faster each year than processing gains. When compounded annually, available bandwidth continues to accelerate and encourages the use of distributed computing in the infrastructure to utilize the 'edges' of the computing infrastructure. We can see this distributed computing trend by simply looking at the history of computing – we've moved from mainframes to client/server to n-tier computing to Web applications and now to Web services.

Alex Andrianopoulos is vice president of product management for SeeBeyond. He is responsible for guiding and managing the product management team in the development and execution of a cohesive product strategy plan, further strengthening SeeBeyond's technological leadership in e-business application integration.
ALEXA@SEEBEYOND.COM
and INFO1@SEEBEYOND.COM

The concept of Web services has certainly taken the technology industry by storm, having now gained as much — if not more — attention as its predecessor in standards-based technology, XML. From the technology-savvy developer to the Wall Street–savvy financial analyst, Web services are being discussed as a silver bullet for solving the persistent challenges of integration. In the midst of all this hype, and in anticipation of the widespread adoption of Web services, the staying power of e-business and application integration (EAI) solutions is constantly being questioned. This articlewill demystify Web service standards and clarify their role within an enterprise integration strategy.

According to "Application Integration Thriving Among Standards" from Gartner in July 2001:

Standards are important, but they will neither eliminate the persistent difficulty of heterogeneity nor eliminate the need to integrate application systems through hard work and robust integration middleware technology.

Web Services 101: No Rocket Science Here

Before an organization makes a considerable investment in utilizing Web services, it must have a clear understanding of what Web services are and what business and technology value they can bring to streamlining operations. So, let's start with a simple definition:

A Web service is an application that adheres to new connectivity standards (SOAP, WSDL, and UDDI) that are based on more mature Internet standards (HTTP and XML). This standards-based connectivity allows a Web service implementation to dynamically discover and interact with other Web services automatically.

"In their simplest form Web services standards enable applications to dynamically advertise their own capabilities, search for other applications on the network, and then invoke their services without prior design or negotiation"

THE LARGEST INTERNATIONAL

XML CONFERENCE & EXPO IN THE WORLD!

XML EDGE conference & expo

JUNE 24-27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

XML-NEXT G OF ENTERPRISE DEPLOYMENT

REGISTER ONLINE TODAY

FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Focus on XML during information-packed days while you enjoy an XML/Web Services Keynote panel, comprehensive conference sessions, and an unparalleled opportunity to exchange ideas with peers and industry leaders.

A Sampling of XML-Focused Sessions

- XML STANDARDS - AN OVERVIEW
- OASIS STANDARDS UPDATE
- UBL - A UNIVERSAL BUSINESS LANGUAGE
- BRINGING XML TO PKI
- LINK MANAGEMENT WITH XLINK
- PRACTICAL XSLT AND XPATH
- ENTERPRISE CONTENT MANAGEMENT WITH XML
- XML IN THE ENTERPRISE AND INTER-ENTERPRISE WORLD

ROBERT MIKULA
XML CHAIR
BOARD OF DIRECTORS, OASIS
INDUSTRY EDITOR
WEB SERVICES JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 I-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...

JEAN-FRANCOIS ABRAMATIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C

TYLER JEWELL
PRINCIPAL TECH. EVANGELIST • BEA

DAVID LITWACK
CEO
SILVERSTREAM

ANNE THOMAS MANES
CTO
SYSTEMET

BARRY MORRIS
CEO
IONA

RICK ROSS
FOUNDER
JAVALOBBY

PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: RICHARD ANDERSON
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3058
RICHARD@SYS-CON.COM

SPONSORED BY:

IONA | END 2 ANYWHERE™
ADDS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake
ALTOVA

MEDIA SPONSORS

Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CF Advisor
WebServicesMall
XML.ORG
WIRE
jCert
wrox
OASIS
XML.ORG
SD Times
JAVA
wireless
XML
WebLogic
WebServices
WebSphere
ColdFusion

OWNED AND PRODUCED BY

SYS-CON MEDIA
SYS-CON EVENTS

So, in its simplest form Web services standards enable applications to dynamically advertise their own capabilities, search for other applications on the network, and then invoke their services without prior design or negotiation. Just as people use Web pages to manually interact with Internet applications, systems will use Web services to automatically interact with Internet Applications.

Web services standards can drive value today by increasing interoperability to lower the cost of integration between systems. Going forward, entirely new value propositions will be created as Web services standards are widely adopted to enable applications to dynamically discover each other on the Internet and exchange data.

Tools of the Trade: Web Services Connectivity Standards

Enabling standardized system connections requires that everyone adopt the same protocols, or sets of rules. The foundation of these protocols is now in place and higher-level protocols are emerging. The promise of Web services' eventual success lies in the widespread adoption of these existing standards.

- ***SOAP (Simple Object Access Protocol):*** A simple XML-based messaging protocol for calling a Web service, typically using HTTP transport
- ***WSDL (Web Services Description Language):*** An XML-based language for documenting what a Web service does, where to find it, and how to access it
- ***UDDI (Universal Description Discovery and Integration):*** A directory standard for registering Web services that users and systems can find and use

As these standards mature over the next decade, they'll have a substantial impact on enterprise infrastructure for several reasons. Foremost, as Web services standardize connectivity between systems, they'll drive interoperability and abstraction that will lower integration costs, increase corporate agility, minimize vendor lock-in, and enable best-of-breed deployment strategies. As they're based on mature standards like HTTP and XML, they're relatively simple for developers to adopt and less costly to deploy. Finally, we know that Web services standards will have an impact, as there is such widespread acceptance and adoption of these new standards among software vendors.

Why Are Web Services Increasingly Important?

To understand the value that Web services can really bring to an organization, one must uncover the real-world business challenges that they can help organizations overcome. Industries such as manufacturing, financial services, telecommunications, and utilities will stand to benefit most immediately from the use of Web services, as they are each faced with fragmented supply chains where standardized connectivity between business partners is currently limited, yet critical for maintaining competitive differentiation.

Enabling real-time responses to changing business conditions, Web services offer implementation simplicity, firewall neutrality, and independence from implementation technology. Web services standards effectively predefine communications between applications to remove the need for new connectivity decisions to be implemented for each project. Using Web services, the interfaces to new applications can be readily reused by other applications in an organization or selectively offered to its business partners.

It's easy to see the impact of these benefits in Supply Chain Management and Customer Relationship Management (CRM) integration solutions. For example, accessing Web services from suppliers for product pricing, availability, and order management, as well as exposing these same functions to customers, can greatly increase the visibility and efficiency of an organization's supply chain. Similarly, within a CRM solution Web services can support easy alerting of order status information to any device, bringing the customer closer by improving the service provided. It's important to note, however, that Web services standards are not necessarily enabling capabilities that weren't possible prior to their existence. Rather, all these example e-business initiatives could be performed using XML over HTTP, but would simply take longer to implement and cost more over time, as there were no assurances of interoperability and reuse.

With these opportunities in mind, we're just starting down the road of the recent Web services movement and the continued evolution toward common connectivity standards. Standards maturity and mainstream adoption are still several years away. However, faced with these emerging standards and varied options, global organizations can take action now to implement a strategic architecture to allow them to leverage existing investments while rapidly taking advantage of future advances.

EAI 101: The Framework Behind Web Services Integration

Web services standards are a welcome enabler for efficient system-to-system connectivity; however, the current standards leave many questions unanswered when it comes to meeting enterprise integration needs, such as data transformation for interaction with complex back-end systems, guaranteed messaging, business process management, trading partner and protocol management, transactional integrity, and security.

"Web Services: Platforms and Tools," a METAGroup presentation in February 2002, said:

> *"Web services ease integration, but their current state does not replace existing EAI functionality. Web services tooling won't solve integration problems any more than Microsoft Word solved writing problems."*

EAI technology provides an open, extensible framework for connecting applications within or between enterprises. EAI solutions ensure compliance and interoperability through open messaging, open queuing, open development tools, adapters (applications, Web, e-commerce, communications, legacy, generic) and data standards (XML, EDI, HL7, Swift) across all major platforms.

As an addition to existing connectivity

options (such as Java Messaging Service, FTP, SMTP, HTTP, RMI, CORBA, and API access), Web services standards offer a valuable new real-time connectivity path to use in integrating systems. EAI frameworks have evolved over the past decade to lower the cost of integrating systems by providing an integration framework that is open to existing and future standards and technologies. The goal is for the integration framework to provide a standards-based integration platform that operates across heterogeneous environments and then insulates the end user's integration work from changes in standards, operating system, application versions, and so on.

Within an EAI framework, Web services integration will exist at both the adapter and the business process level. Adapters provide communications and data transformation capabilities to enabling rapid integration into packaged, legacy and custom applications. Any of these applications can in turn access a Web service or be called by a Web service through a SOAP Messaging Adapter to Web service–enableing these existing applications. The adapter performs the hard work by mapping the existing system into the Web service interface.

At the business process level, Business Process Management (BPM) solutions are used to control the flow of business process activities across the systems and users involved. BPM solutions typically offer process modeling, implementation, monitoring, management, and process optimization services to streamline processes and improve efficiency. Web services can be called as needed from any activity within a business process, and in turn, a larger business process can be exposed as a Web service to be called by other systems or partners. The value of Web services in this context is in standardizing the interfaces in and out of the business process.

The Future of EAI and Web Services

Every organization's technology infrastructure is a complex and heterogeneous environment, and the vision of a comprehensive end-to-end e-Business solution based entirely on standards that are universally supported on every computing platform is far from reality. Following the earlier hype cycle driven by XML, organizations realized that although XML is a very effective format for exchanging data, it is not the panacea that was promised for all integration challenges .

Faced with today's rapidly evolving standards and ever-increasing demands for business agility, global organizations require a strategic architecture that allows them to both leverage existing investments and to take advantage of future advances. Web services offer a nondisruptive model for extending and enhancing existing applications without forcing any wholesale replacements. EAI proved to be the top IT spending priority among CIOs, as noted in the February 2002 Morgan Stanley independent CIO survey. With that momentum, it continues to allow information to work harder and smarter, increasing the speed of business reaction time and facilitating seamless, straight-through transaction processing.

Web services will similarly drive great efficiencies across deployments by removing unnecessary connectivity friction between implementations, but this is not where the greater challenges lie. Creating and managing flexible integrations between systems that deliver value to the enterprise is the hard work, and requires robust application integration capability, trading partner/protocol management, and business process management all running securely on a scalable, reliable, distributed, yet centrally managed, platform.

EAI platforms have adopted these new systems-connectivity standards to lower integration costs for customers, and will continue to both guide and adopt new standards for systems connectivity and integration as they are developed. Contrary to the positioning of some people that Web services will replace EAI software, the reality is that EAI will simply continue to evolve to lower the cost of integrating systems.

FOCUS ON EAI

written by Matjaz B. Juric
and Marjan Hericko

J2EE, EAI, & Web Services

New challenges for your information systems

Today Web services are believed to be the crucial technology for e-business. Technically, they don't differ considerably from distributed components, such as EJB (Enterprise JavaBeans), CORBA (Common Object Request Broker Architecture), or even COM+ (Component Object Model). Web services have:

- An XML-based transport protocol: SOAP (Simple Object Access Protocol)
- An XML-based language that defines the interfaces of Web services: WSDL (Web Services Definition Language)
- A directory service: UDDI (Universal Description, Discovery, and Integration)

Matjaz B. Juric is an assistant professor at the University of Maribor. His PhD work received awards from the Slovenian IEEE Section and he participated the OOPSLA Doctoral Symposium. He has been involved in several large-scale projects with object technology, including the development of RMI-IIOP. Matjaz is the coeditor of the upcoming special issue of the *International Journal Informatica* on component-based software development.
MATJAZ.JURIC@UNI-MB.SI

Marjan Hericko, PhD, is an assistant professor at the University of Maribor. He has been involved in several large-scale projects with object technology and is an expert in object design, EAI, software quality, and component development.
MARJAN.HERICKO@UNI-MB.SI

The mission of Web services however, is to provide high-level interoperability interfaces for business collaboration. Web services are based on XML technologies, designed from the ground up for seamless interoperability over the Internet. Therefore, Web services should be designed as document-oriented exchange systems and tuned to the needs of business collaborations. This differs from the object-oriented interfaces of business-tier components. Business-tier components require tighter coupling, which is difficult to achieve within companies on shared business processes. Web services, on the other hand, connect independent companies (actually their information systems) in a more loosely coupled fashion.

Web services pose new challenges for an information system. Today's requirements are very high – companies need to provide instant, online access to up-to-date information delivered with efficiency, reliability, and quality. Customers expect immediate response, and are not satisfied with days (nor even hours) of delays in confirming orders.

However, these delays are often the case when e-business is not backed by an efficiently integrated enterprise information system (EIS). In most cases, Web services will not implement their functionality themselves. Rather, they'll make several calls to the EIS applications. The problem here is that existing EIS applications have usually been developed with different, even legacy, technologies, and are very heterogeneous. Accessing these systems directly from Web services can be difficult because each existing system is unique and potentially requires a unique way to access it. This approach requires far too large an effort for development, and particularly for maintenance, because each change in the existing system will require an update in all related Web services.

Therefore, enterprise application integration (EAI) is the most important prerequisite for efficient Web services. Only an integrated information system inside the company allows on-demand processing of e-business requests. The responsiveness of Web services and the adequate quality of information provided through them can only be achieved by the tight integration of an enterprise information system in the back end. EAI is the key success factor for the successful introduction of Web services.

Research from leading consulting companies such as the Gartner Group confirms this thesis. It also shows that today there are very few Web services (or other means for front-end systems to directly communicate with business partners) that are efficiently integrated with the back-end applications inside the company. Most nonintegrated solutions fail to meet business expectations.

THE LARGEST INTERNATIONAL

WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

web services EDGE conference&expo

JUNE 24-27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

WEB SERVICES
SKILLS, STRATEGY, AND VISION

REGISTER ONLINE TODAY

FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.

A Sampling of Web Services-Focused Sessions

- PRACTICAL EXPERIENCES WITH WEB SERVICES AND J2EE
- STATE OF THE WEB SERVICES INDUSTRY
- THE ODD COUPLE: MAKING .NET AND J2EE WORK TOGETHER
- EXPLORING THE .NET MY SERVICES INITIATIVE
- STANDARDS WATCH
- GUARDING THE CASTLE: SECURITY & WEB SERVICES

SEAN RHODY
CONFERENCE TECH CHAIR
WEB SERVICES TRACK CHAIR
EDITOR-IN-CHIEF
JDJ SERVICES JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 I-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

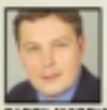
Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...

JEAN-FRANCOIS ARNAUDIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C

DAVE CHAPPELL
VP CHIEF TECHNOLOGY EVANGELIST
SONIC SOFTWARE

GREGG KIESSLING
CEO
SITRAKA

ANNE THOMAS MANES
CTO
SYSTINET



BARRY MORRIS
CEO
IONA

ERIC RUDDER
SENIOR VP DEVELOPER AND PLATFORM EVANGELISM
MICROSOFT

PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3057
MICHAEL@SYS-CON.COM

SPONSORED BY:

IONA | END 2 ANYWHERE™
ADDS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake
ALTOVA

MEDIA SPONSORS

Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CE Advisor
WebServicesMall
XML.ORG
WIRE
jCert
wrox
BASIS
SD Times
JAVA
wireless
XML
WebLogic
WebServices
WebSphere
ColdFusion

OWNED AND PRODUCED BY

SYS-CON MEDIA
SYS-CON EVENTS

What is Enterprise Application Integration?

EAI, although not a buzzword anymore, is far from being an easy or well-defined task. Indeed, it has become one of the most difficult problems facing enterprise application development in the last few years. The major challenges relate to the integration of the different domains, architectures, and technologies. If we wish to integrate, we must also find ways to allow the coexistence of different architectures, and solve problems inherent with integrating several different technologies. In addition to all of this, we also have to look closely at application content in order to solve the problems of data redundancy and different views of the same problem. These conceptual challenges may be more difficult to overcome than the technical ones. To make things even more difficult, there is often a significant investment already in place for a variety of application integration technologies that we must deal with. Furthermore, the requirements placed on information systems are both growing significantly and changing frequently. Integration projects, therefore, have to be performed in the shortest possible time, deliver results quickly, and adapt to these ever-changing requirements. Of course, the resources for integration are often limited, too.

EAI is the total integration of applications within an enterprise. From the business perspective, it is the competitive advantage a company gets when all applications are integrated into a unified information system capable of sharing information and supporting business workflows. Information must often be gathered from several domains and integrated into a business process. Without EAI, although the required information may well be available and exist in some form somewhere in an application, for typical users it is practically impossible to access it online.

From the technical perspective, EAI refers to the process of integrating different applications and data to enable sharing of data and integration of business processes among applications without having to modify these existing applications (too much). EAI must be performed using methods and activities that enable it to be effective in terms of costs and time.

EAI with J2EE

In most real-world cases, companies have to perform EAI together with the development of new solutions (including Web services). Then it becomes a logical step to consider a modern software platform for both, and the new development. Java 2 Enterprise Edition (J2EE) is one of the two most important modern software platforms today. The other is Microsoft .NET. In this article, we will focus on J2EE.

As we know, J2EE is a very suitable platform for the development of Web services. Let's consider whether J2EE is also suitable for EAI. For successful EAI we need middleware technologies that provide an adequate infrastructure for the communication between existing applications. The J2EE platform provides support for a number of middleware APIs and protocols. The most important middleware APIs in J2EE are JDBC, RMI-IIOP, Java IDL, JMS, Java Connector Architecture, and JNDI. Very important APIs, which we'll discuss later, are JAXP (and other JAX APIs) for XML support, JTA and JTS for transaction support, and JAAS for authentication and authorization. The most important protocols supported by J2EE are HTTP(S), and IIOP.

For successful integration, we also have to define the integration architecture. This architecture should take advantage of the existing system's information and code and should easily accommodate new generation applications, particularly e-business solutions such as Web services.

Therefore, the integration architecture should be similar to the architecture of modern applications. This means that it should be multitier and component-based. The problem, however, is that most existing and legacy systems don't comply with this architecture. They don't look like components and they're not multitier. The challenge, therefore, becomes how to make the existing systems look like modern ones.

The solution is to place existing applications in the data-persistence or EIS tier and

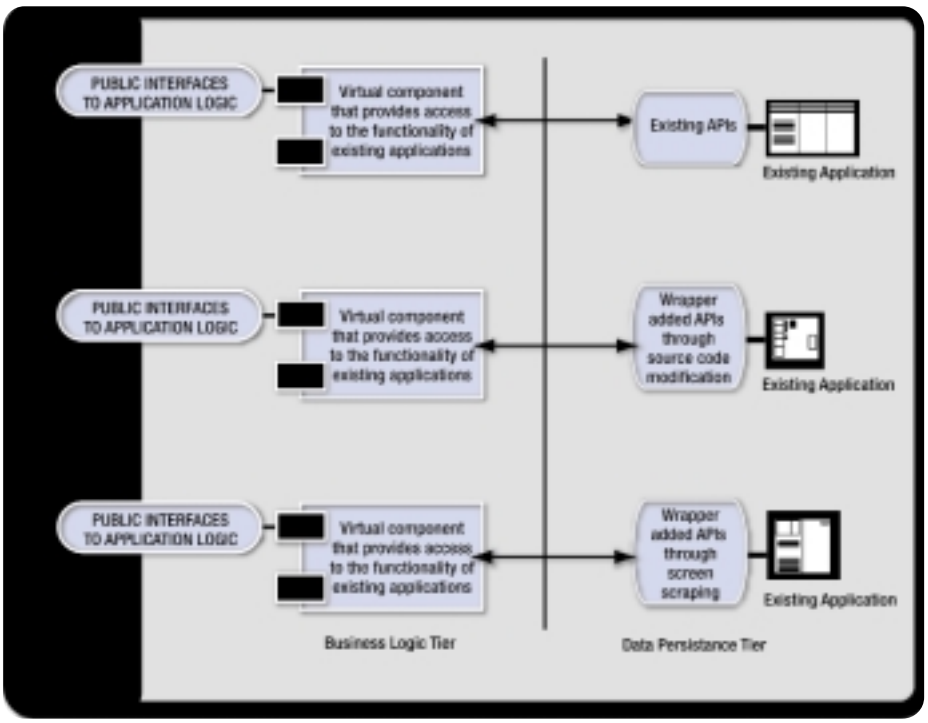


FIGURE 1 **Virtual components provide access to existing applications**

to find ways to encapsulate these existing applications into virtual reusable components that will expose the functionality though interfaces. Clients accessing these components through the interfaces won't see whether they're dealing with newly developed components or encapsulated legacy components; they'll access both in the same manner.

Virtual components implement the facade pattern for the existing systems. However, to provide the functionality of existing systems, virtual components have to access them programmatically. This can be achieved relatively simply if existing applications already provide some APIs. Otherwise, or if they're provided but don't meet our requirements, then we'll have to define and build them. For this, we'll modify the existing application and add the necessary APIs. This is called wrapping and is shown in Figure 1.

Adding wrappers can be quite straightforward, especially in solutions that have been developed in-house and where source code and documentation are available. However, with applications where we don't have the source code and where the documentation is limited, changing the application can be difficult.

The most important fact is that the component interfaces become the contracts in the integrated architecture. From that perspective, the interfaces have a very important role. As long as they stay unchanged, we can modify the implementation of the components without influencing the rest of the system. In other words, as long as we use the same interfaces, we can replace existing systems with newly developed solutions and none of the clients will ever know that a change has occurred.

Therefore, how we define the virtual components is very important. We have to focus on their semantics and define them on a highly abstract level, rather than just exposing the functionality of existing applications. Otherwise, it would be very difficult to replace existing systems with the new ones and preserve the interface. Con-

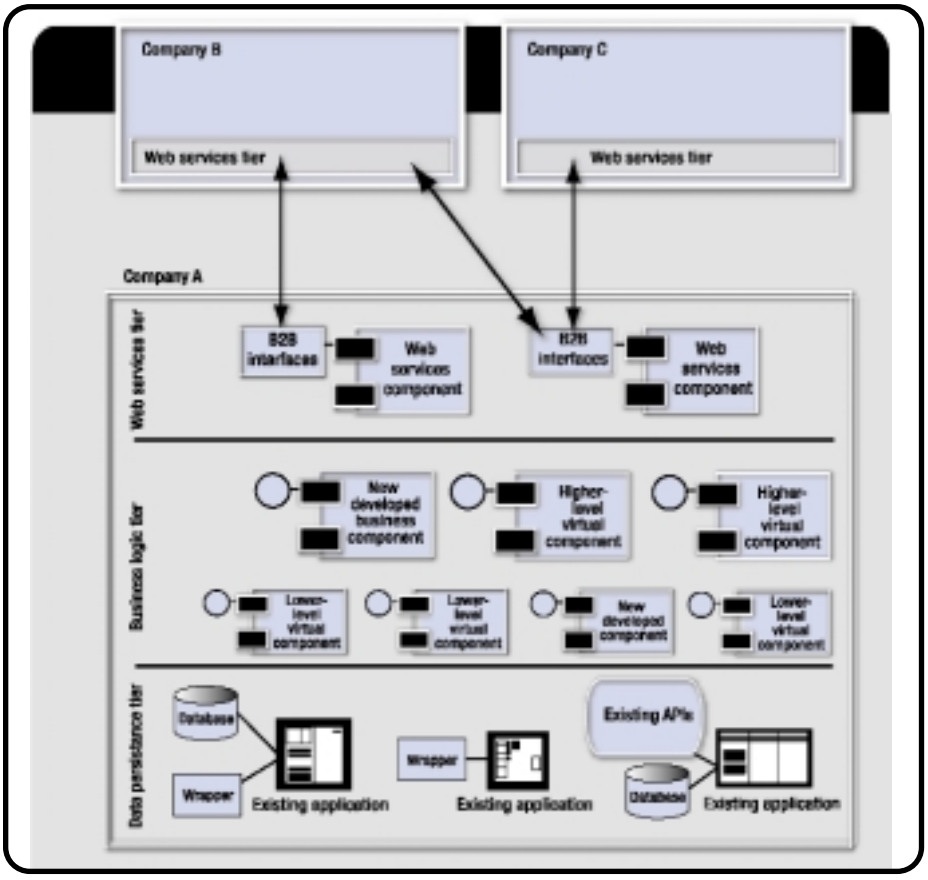


FIGURE 2 **Web services built on top of integrated information system**

sequently, achieving efficient integration is always connected with redesigning the information system as a whole, which is also a good opportunity to perform business process reengineering.

J2EE Technologies for EAI

J2EE provides the necessary technologies for developing virtual components and wrappers. Because J2EE supports several open standards for communication between tiers, such as HTTP(S) and IIOP protocols and APIs, such as JDBC, JMS, JNDI, and JCA, we can include components that have not been developed in Java on practically any of the tiers, thus providing a way to integrate existing applications.

On the business-logic tier we can deploy not only EJB components, but also CORBA (and RMI-IIOP) distributed objects and components communicating through a JMS-compliant MOM. This is one of the most important facts for integration. We can see that for developing virtual components on the business logic tier, we're not limited to EJBs but can use CORBA, RMI-IIOP and MOM components too. Please note that CORBA and MOM components don't have to be developed in Java; rather, we can use a variety of programming languages. Both CORBA and leading MOM products support all popular programming languages, including C++, C, Smalltalk, Ada, COBOL, Delphi, and even Visual Basic.

These non-Java components don't reside inside the EJB container, and as such they can't take advantage of the managed environment provided by containers. However, they can still take part in transaction and security mechanisms. For this, they have to use the corresponding APIs. Support for both–transactions, and security–is crucial for successful EAI. Let's look at how transactional and security integration can be achieved in J2EE.

To access the J2EE transaction service functionality, we can use JTA (the Java Transaction API). JTA provides an abstraction layer on top of JTS (the Java Transaction Service). This abstraction layer enables the platform to choose which JTS implementation it will use. The implementation, which is typically provided by the application server, is transparent to the application components because those components don't interact with JTS directly.

The really important fact for integration is that JTS is compliant with CORBA Object Transaction Service (OTS) 1.1. More exactly, JTS is the mapping of the CORBA OTS 1.1 to Java. It propagates transactions using the IIOP protocol. CORBA OTS provides support for distributed transactions and interoperability with X/Open DTP (Distributed Transaction Processing)–compliant transactional systems. Fortunately, the majority of today's transactional systems are X/Open DTP compliant.

However, the J2EE specification 1.3 *does not require* a J2EE implementation to support distributed transactions. Fortunately, most important J2EE application servers, such as BEA WebLogic, IBM WebSphere, Oracle 9*i*AS, and others, provide support for distributed transactions. If we choose such application servers, we'll be able to achieve transactional interoperability relatively easily with X/Open DTP–compliant EIS systems.

EIS systems that implement their own local transactions, which are not X/Open DTP–compliant, will require more manual work to achieve transaction integration. We'll have to commit or roll back each system explicitly and manually. One technique to manage this scenario is to use compensating transactions.

Security is another important topic for EAI. As we know, Java has a policy-driven, domain-based security model. It provides extension packages, which were initially introduced as optional but have been integrated into J2SE 1.4, including:

- ***Java Authentication and Authorization Service (JAAS):*** Provides a framework for authentication and authorization. J2EE 1.3 requires that J2EE application servers use JAAS in Web components and EJB containers, particularly to support J2EE

THE LARGEST INDEPENDENT

JAVA DEVELOPER CONFERENCE IN THE WORLD!

JDJ EDGE conference&expo

JUNE 24-27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

JAVA IN JUNE
ESPECIALLY IN NEW YORK

REGISTER ONLINE TODAY
FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to i-technology professionals and senior IT/IS strategy decision makers.

The Java Fast Picks on June 24, a Java-focused CEO Keynote Panel, and comprehensive conference sessions will focus you on Java-busy information-packed day!

A Sampling of Java-Focused Sessions

- JAVA 1.4: WHAT'S NEW?
- BUILDING TRULY PORTABLE J2EE APPLICATIONS
- JAVA TOOLS FOR EXTREME PROGRAMMING
- BUILDING ASYNCHRONOUS APPLICATIONS USING JAVA MESSAGING
- .NET VS. J2EE
- J2EE: SETTING UP THE DEVELOPMENT ENVIRONMENT
- BUILDING WEB SERVICES WITH J2EE
- DETECTING, DIAGNOSING, AND OVERCOMING THE FIVE MOST COMMON J2EE APPLICATION PERFORMANCE OBSTACLES

ALAN WILLIAMSON
JAVA CHAIR • EDITOR-IN-CHIEF
JAVA DEVELOPER'S JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 i-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- i-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...

TYLER JEWELL
PRINCIPAL TECH. EVANGELIST • BEA

GREGG KIESSLING
CEO
SITRAKA

DAVID LITWACK
CEO
SILVERSTREAM

BARRY MORRIS
CEO
IONA

GREGG O'CONNOR
PRESIDENT
SONIC SOFTWARE

RICK ROSS
FOUNDER
JAVALOBBY

JAMES DUNCAN DAVIDSON
FATHER OF ANT

For Exhibit Information

CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3057
MICHAEL@SYS-CON.COM

SPONSORED BY:

IONA | END 2 ANYWHERE™
ADDS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake
ALTOVA

MEDIA SPONSORS

Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CF Advisor
WebServicesMall
XML.ORG
WIRE
jCert
wrox
BASIS
SD Times
JAVA
wireless
XML
WebLogic
WebServices
WebSphere
ColdFusion

OWNED AND PRODUCED BY

SYS-CON MEDIA
SYS-CON EVENTS

Connector Architecture.

- ***Java Secure Socket Extension (JSSE):*** Provides packages for enabling secure communications using SSL and TLS protocols and includes functionality for encryption, authentication, and message integrity.
- ***Java Cryptography Extension (JCE):*** Provides a framework and implementation for encryption, key generation, key agreement, and MAC (Message Authentication Code) algorithms.

For authentication, JAAS supports different models to be plugged in at runtime. This enables us to use an industry-standard authentication technology through all the integrated applications. The most commonly used authentication technology that enables single sign–on support for multiple applications is Kerberos. Kerberos is a network authentication service created by MIT. For more information on Kerberos see http://web.mit.edu/kerberos/www.

JAAS, together with JSSE and JCE, enables us to build our own EAI security infrastructure. It allows us to wrap the existing applications in the same security process – using industry standards such as Kerberos for authentication and SSL and TLS for communication channel protection and application-independent encryption engines.

In EAI projects that involve integration of non-Java applications and resources, J2EE-based security may not always be sufficient to achieve end-to-end security. An alternative is to use the CORBA security model. This model can be used from Java as well as from applications written in other programming languages. It's outside the scope of this article to go into the details of CORBA security. More information can be found at www.omg.org.

Another possibility is to use the Generic Security Services (GSS) API, developed by the Internet Engineering Task Force. GSS provides a generic authentication and secure messaging interface that supports pluggable security mechanisms. GSS version 2 is defined in a programming language–independent format. Through GSS, applications can use different security mechanisms without having to make changes to the application itself. Java bindings for GSS are referred to as JGSS. The JGSS API is part of J2SE 1.4. JGSS allows Java developers to create uniform access to security services over different mechanisms, including Kerberos.

JGSS has some important features that the Java security model doesn't, including support for Kerberos cipher suite; ability to use any transport protocol for communication, not just sockets; and selection of encryption type. JGSS is likely to emerge as the standard API for security integration in EAI.

Support for transactions and security is the main reason we don't use Web services technologies for developing the EAI architecture. Web services currently don't provide adequate support for transactions and security and they have other disadvantages for EAI. Their performance and scalability is lower due to the use of XML-based protocols, which require more effort for marshaling and generate more network traffic. Finally, it's important to note that the tools and APIs for Web services in J2EE are still under development.

Web Services – the Next Logical Step

We can build Web services directly on top of the proposed EAI architecture. We should consider them as highly interoperable entry points for business partners. One approach is to map the business logic component interfaces to Web services directly, which will usually require that we develop a dedicated component (for example an EJB) that will provide a suitable document oriented interface. This approach is supported by the leading J2EE application servers, which provide tools to convert EJBs to Web services.

The other approach is to build Web services on workflow-based models, where each Web service calls several business components and other resources to fulfill the request. This approach requires more development work and some knowledge of the corresponding APIs. J2EE version 1.3 currently provides support for XML through the Java API for XML Processing (JAXP) only. Support for XML, provided through JAXP, is very primitive if we want to use it for Web services development. Fortunately, Sun's Java Community Process is currently defining additional XML-related specifications, most of them focused on Web services, including: JAXM (JSR 67) Java API for XML Messaging, JAX/RPC (JSR 101) Java API for XML-based Remote Procedure Calls, JAXR (JSR 93) Java API for XML Registries, JWSDL (JSR 110) Java Web Service Definition Language, and JAXB (JSR 31) Java API for XML Binding. These APIs are still under development and not a standard part of J2EE 1.3. In the meantime, we have to use custom APIs provided by application server vendors or add-ons, which can limit the portability in the future. Therefore, the first approach seems to be more suitable today. Figure 2 shows how Web services can be integrated on top of the composite information system EAI architecture.

Conclusion

We've seen that Enterprise Application Integration (EAI) is crucial for developing efficient and well-connected Web services. Only if the Web services are satisfactorily connected with back-end information systems can they provide immediate and accurate information to business partners – a prerequisite for successful e-business collaboration.

The fact is that most Web services will have to use existing and legacy applications as back-end solutions. Making the integration between such systems efficient will be the key success factor. In particular, immediate response and immediate propagation of data to all related applications will be the major issue. Web services that aren't efficiently supported by back-end applications will most likely fail to meet the requirements.

Resource

- Juric, Matjaz B. S. Jeelani Basha, Rick Leander, and Ramesh Nagappan (2001). *Professional J2EE EAI* (Wrox Press)

FOCUS ON EAI

written by Carol Murphy

Will Web Services Mean the End for EAI?

Complementary approaches for peaceful Coexistence

Increasing visibility throughout the supply chain, improving efficiency across the enterprise, responding to regulatory or competitive pressures to reduce cycle times, eliminating errors due to inaccurate or out-of-date information, collaborating with your business partners. The common thread across each of these is the need for dissimilar business applications to interoperate effectively in support of business goals.

Carol Murphy is a partner and the solution architect for CSC Consulting's Enterprise Application Integration (EAI) practice. Carol educates clients on using EAI technology for business integration, including vendor selection, project planning, and implementation of best practices. She is the primary author of the CSC *Business Integration Practice Guide*.
CMURPH23@CSC.COM

In any large organization, managing and maintaining the often spaghetti-like maze of connections between applications – your own and those of your business partners – is a costly and perpetual challenge. Enterprise application integration (EAI) is an effective solution. It offers a business process– oriented approach to integration that can insulate the processes that are the business from the systems that run it, facilitating a rapid response to changing business conditions. EAI creates efficient enterprise-wide communications, making information easily accessible throughout what Gartner calls the "enterprise nervous system." Most important, EAI reduces IT development and maintenance costs by minimizing expensive (and often redundant) custom development of point-to-point application connections. Note that in this article, I use the term EAI to cover both intraenterprise and interenterprise (e.g., A2A and B2B) application integration.

Web services are also predicted to transform the development and deployment of applications throughout and between enterprises. Some advocates even claim that Web services will spell the demise of EAI. Only time will tell whether this will be the case, of course. In this article, we'll examine EAI and Web services architectures and explore how each might be used within your enterprise. While there are similarities between the two models, significant areas of differentiation still remain that make EAI and Web services more complementary than competitive.

Enterprise Application Integration Architecture

EAI builds on proven middleware techniques such as message brokering and data transformation, and introduces architectural components called adapters for communication with applications and other data sources. EAI also incorporates business process modeling and workflow, metadata management, security and system administration, and monitoring. Working in concert, this set of services provides a robust environment for integrating disparate applications within and across enterprises.

> "The choice between EAI and Web services does not need to be an either-or proposition"

Message Brokering

A message broker handles the fundamental role of ensuring reliable communication between applications and other system components. It manages the complex details of communication among different hardware platforms, operating systems, and network transport protocols. The message broker typically supports various service levels such as guaranteed once-only delivery or return receipt. It also supports both synchronous and asynchronous communication models such as publish-subscribe, publish-reply, and request-reply. The message broker uses rules, generally based on message type or message content, to route messages for delivery to appropriate recipients. Most important, if the recipient is unavailable for some reason (e.g., the receiving application is not running, the receiving system is unavailable, or there's some type of network difficulty), the message broker will queue the message(s) for future delivery once the recipient becomes available.

Data Transformation

Transformation services are essential to application integration since applications rarely, if ever, share a common data format or semantic model. This is even more apparent with business-to-business communication, even with all the standards-based efforts around common interchange formats such as EDI, RosettaNet, or ebXML. Traditionally, each individual application embeds separate transformation logic in its code base for every other application with which it integrates, thus dramatically increasing both application development and maintenance costs. EAI transformation services handle conversions among the differing message formats and semantic models used by the various applications participating in the EAI environment. Typically, EAI vendors offer a graphical interface and processing engine to create transformation and mapping rules, which can range from simple one-to-one field mapping to complex fan-in/fan-out operations involving external table lookup or multi level conditional substitution. With appropriate forethought during the design phase, transformation rules can often be reused for multiple integration scenarios, further reducing development and maintenance costs.

Adapter Architecture

An EAI system uses adapters to expose the functionality – in other words, the services – offered by participating applications or other data sources in a consistent manner. An adapter manages communication between the EAI system and the application, translating EAI system messages into application-specific events (and vice versa). In some cases, an adapter may translate a single EAI message into multiple calls to the application's programming interface, thus providing a higher-level "service interface" for the application. Adapters are typically designed to be non invasive – that is, to use whatever interface mechanisms are already supported by the application so it requires no modification in order to communicate with the EAI system. Adapters can also handle management functions such as alerting the EAI system if the participating application becomes unresponsive or unavailable.

All EAI vendors offer an adapter development kit to create your own adapters. Most also offer a wide range of prebuilt adapters for commercial applications or other technologies, which can dramatically reduce development time. Examples include:

- ***ERP and CRM applications:*** SAP, Oracle, PeopleSoft, and Siebel
- ***Application-independent formats:*** Flat file, fax, e-mail, or wireless application protocol
- ***Database protocols:*** ODBC, JDBC, DB2, and SQL
- ***Internet formats:*** HTTP/HTTPS, XML, WS DL, and SOAP
- ***Custom applications:*** Developed using COM/DCOM, CORBA, and EJB
- ***Mainframe protocols:*** IMSL and CICS
- ***Industry-specific formats:*** EDI/EDI FACT, SWIFT, HL7, and RosettaNet

- ***Messaging systems:*** MQSeries, TIBCO, and JMS

Business Process Modeling and Management

Sophisticated EAI packages now offer a business process–centric model. During the application design phase, business processes are modeled as a set of integrated, event-driven activities (or nested processes) supported by multiple applications or people. These integrated business processes may include activities that span departments, divisions, or enterprises. People may also be involved if the process requires an approval or override or if the situation requires a judgment call (e.g., deciding whether to underwrite a questionable insurance risk). During the development phase, the integrated business process models are mapped to a set of messages, information objects, transformation, and other processing rules, and supporting application or technology adapters. In production, the EAI software "runs" the integrated business process models, orchestrating the event-driven information flow between applications. If an exceptional condition causes a process to terminate or stall, the EAI system can alert the system operator or monitoring software to the nature of the problem.

Many integrated business processes are composed of multiple activities, some of which must be completed as a "work unit" in order for the process to be considered successful. While the traditional two-phase commit approach can't be applied to loosely coupled system integration, where applications may span enterprises and transactions may last for hours or days, the EAI system does need to ensure that all steps in a business transaction are successfully completed before allowing the entire transaction to progress forward. In the case of failure, the EAI system can restart the transaction at a known "stable" point. In some cases, a designer can even teach the EAI system to take corrective action by defining compensating transactions within an integrated business process that can be invoked to "undo" or work around a problem.

Metadata Repository

The EAI metadata repository acts, in essence, as the service directory for all information about the integration environment. Metadata includes definitions for integrated business process models, messages and integration objects, rules governing data transformation and other processing, parameters for application connectivity and system management, user roles and security information, trading partner profiles, and status information about in-flight business processes or other system components. Components within an EAI environment query the repository to determine the services that are available and their connectivity details (note that these services are restricted, however, to the particular EAI environment and not publicly available, as in the Web services model). Changes to system metadata can be automatically propagated to the affected EAI system components. The EAI repository is generally built on an industry-standard database, which may be deployed as a single instance or distributed across multiples sites for better performance.

Security

In an open applications environment, particularly for business-to-business transactions but even for intraenterprise integration in environments such as finance or health care, security concerns are paramount. Security services include authentication, authorization, encryption, nonrepudiation and dynamic setup and closure of sessions between trusted partner companies, where sessions are supported over days rather than seconds. Some EAI vendors support LDAP or other directory services that provide user authentication or authorization information. In an integrated environment, it's important that security be managed across the entire life cycle of an integrated business process, not simply between any pair of activities participating within it.

System Administration and Monitoring

As there are many "moving parts" in an integrated applications environment, system administration and monitoring services are critical. System administration and monitoring services enable an administrator to deploy, manage and monitor system components, including in-flight integrated business processes, message brokers, queues, adapters, and metadata repositories. Some EAI systems can integrate with popular third-party systems' monitoring packages, allowing a system administrator to leverage existing resources and skills and have a unified view of the entire hardware, software, network, and applications and integration environment.

Web Services Architecture

The Web services architecture is defined by a set of specifications covering three major functional areas: service definition, messaging, and service discovery. Future specifications are expected to address other areas such as business process management.

Service Definition

The Web Services Description Language (WSDL) is an XML format for describing network services. WSDL includes both an abstract and a concrete description of a Web service. The abstract definition describes the operations and parameters (i.e., the messages) supported by the service. The concrete definition defines the implementation (or bindings) for the service, including the required network protocols, wire protocols and network endpoint addresses needed to invoke the service.

WSDL does not address any business process–modeling aspects of Web services – in other words, how developers should combine and orchestrate Web services (and possibly other application functionality) to create higher-level integrated business processes, which could themselves then be offered as Web services. Several alternate standardization efforts are under way to address these issues, including Business Process Modeling Language (BPML) and Web Services Flow Language (WSFL).

Messaging

The Simple Object Access Protocol (SOAP) is an XML-based protocol for exchanging information in an open network environment. SOAP describes the structure for a message and how to process it, a set of encoding rules for the message parameters, and a convention for representing remote procedure calls and responses. In the Web services runtime environment, a SOAP "listener" receives and accepts a SOAP message, extracts the XML message body, transforms the message into a native protocol, and delegates the request to the appropriate application for processing.

In theory, SOAP can support almost any

Dynamic Buyer

www.ibm.com/smallbusiness/dynamicbuyer

message transport protocol, although SOAP V1.1 only defines bindings for HTTP. This presents some significant limitations for use in enterprise application integration scenarios, since HTTP does not support reliable or asynchronous messaging and its security capabilities, even using HTTPS and SSL, are restricted to connection-level requirements.

Service Discovery

WSDL is helpful to describe and invoke a Web service and SOAP is helpful for communicating with it, yet both are of limited use if no one knows about the service in the first place. To this end, a Web service provider can advertise its capabilities in a repository conforming to the Universal Description, Discovery, and Interaction (UDDI) standard. Web service clients can query the repository, searching either by service capability (yellow pages model) or by known service provider (white or green pages model). When a match is found, the repository returns the WSDL description corresponding to the Web service, which the client can use to format a request to the service for execution.

UDDI in its current form does have some limitations: it doesn't support the ability to rate service providers in terms of reliability or trustworthiness, to locate alternate service providers, or to notify clients if a service's definition or location has changed. Strictly speaking, UDDI isn't part of the Web services architecture or even necessary to implement it. Web service definitions can be advertised on regular Web pages or stored in conventional system directories. Alternative directory standards have also been proposed that may address these shortcomings.

Peaceful Coexistence

The Web services architecture presents a model for the development and deployment of new network-based application services. The EAI architecture presents a model for integration of new and existing applications, including Web services, within and between enterprises. Although the implementation details still vary considerably, as we've discussed, both EAI and Web services include messaging, service definition and service discovery (i.e., metadata in an EAI world). The Web services model does not currently address business process management, data transformation, system management, security, asynchronous communication, or reliable messaging – which are all essential in an application integration environment.

> **"The main benefit from exposing any (new or existing) application functionality as a service is to allow it to be combined with other services in support of some integrated business process"**

Let's explore a few areas for using Web services and EAI, as shown in Table 1:

- ***Exposing existing application functionality***

Some commercial software vendors, such as SAP, have announced that their products will eventually offer native Web services interfaces, so you could just decide to wait a while. For other existing applications that already support service-oriented interfaces, some EAI vendors offer tools to generate Web service definitions from existing application interfaces, including applications for which they already provide adapters. After all, a Web service definition is simply another form of application interface. On the other hand, if the application does not currently provide a service-oriented interface – for instance, if it requires multiple calls and/or preprocessing logic in order to act service-like – an adapter could also be used to provide a higher-level service interface that hides these details from clients of the service. Finally, if the application does not offer an interface that can be easily exploited, the case for many legacy applications, then exposing its functionality will be a challenge in a Web services or an EAI environment. Leveraging a prebuilt EAI adapter is the best choice in this case.

TABLE 1

Functionality offered in each architecture	Enterprise Application Integration					
	Web Services					
Functionality Required → / Usage Required	Services Definition	Messaging	Service Discovery	Data Transformation	Business Process Management	Manageability
Exposing existing application functionality	✓	✓	✓	✓		✓
Exposing new application functionality	✓	✓	✓			✓
Combining multiple applications into an integrated business process (composite application)	✓	✓	✓	✓	✓	✓
Exposing an integrated business process	✓		✓		✓	✓
Managing a heterogeneous environment	✓	✓	✓		✓	✓

- ***Exposing new application functionality***

It makes sense to consider adopting a Web services model for new application development if you can work within the constraints of the current specifications and recognize that the specifications are still in flux. Since the current Web services model provides no support for reliable messaging, asynchronous communication or stringent security, Web services applications should be limited to internal enterprise usage where these may be less critical. If you do require this functionality, either leverage a pre-built EAI adapter or develop a custom EAI adapter for your application.

- ***Incorporating services from multiple applications into an integrated business process***

The main benefit of exposing any (new or existing) application functionality as a service is to allow it to be combined with other services in support of some integrated business process. This process may itself be considered a composite application, in which loosely coupled applications work in concert to solve a business need. To do this clearly requires the business process management, reliable messaging, data transformation, application connectivity, security and system management capabilities available today only in EAI environments. Of course, you could build these capabilities from scratch as part of a custom application development effort, but one would have to question such an investment decision when robust commercial integration packages are already available.

- ***Exposing integrated business process functionality as a service***

Building on the previous example, many enterprises will want to expose some of their integrated business processes as Web services for internal or external use. EAI tools that can create Web service definitions for integrated business processes will be the best option to provide this capability.

- ***Managing an integrated systems environment***

Enterprise environments include a complex combination of commercial, custom and legacy applications, integrated business processes, and now Web services, along with the systems that support them (e.g. hardware, networks, printers, firewalls). Components within such an environment must be "manageable," and we need tools that can effectively and coherently manage these disparate resources as an integrated whole. This remains a challenge for both Web services and EAI environments, although at least one EAI vendor is working aggressively with leading system management vendors to address it.

Conclusion

It should now be clear that the choice between EAI and Web services does not need to be an either-or proposition. The Web services model is a good choice for new application development, if you can work within the constraints of the current specifications. For integration-related activities that require industrial-strength reliability, security, and manageability, the enterprise application integration model remains the clear choice. EAI vendors are quickly moving to embrace Web services within their own products, so a combination approach is also a viable alternative.

Some day the two models may converge. Even if this does occur, it will still be several years away given the relatively slow pace of standardization efforts and the typical delay before approved standards actually materialize in commercial products. A more likely scenario is that EAI and Web services will peacefully coexist as complementary approaches to the perennial (yet separate) challenges of application development and integration. These are fundamentally different problems, and as every enterprise architect knows, few tools or approaches work well for every situation.

Written by Norbert Mlkula

Quintessent Web Services

A real-world story from the telecommunications industry

Dave Deutschman,
CTO, Quintessent Communications

WSJ's Industry Editor, Norbert Mikula, recently spoke with Dave Deutschman, chief technology officer of Quintessent Communications, headquartered in Seattle, WA., about their entry into the Web services market.

Dave, can you tell us about Quintessent and your duties as CTO and VP of Marketing?

Quintessent Communications, Inc. (QCI) is the leading provider of interconnection gateway solutions for telecommunications companies. These solutions form the basis of carrier-to-carrier provisioning of telecommunications services such as voice, data, wireless and cable. In essence, Quintessent provides supply chain automation software for the communications industry. QCI's solutions manage the business rules required to exchange service order data, allowing carriers to dramatically improve the efficiency of provisioning new orders and thereby significantly improving return on investment of their networks.

My role at Quintessent is to look for new opportunities where we can apply our core products and competencies to create new products and increase revenues. I am also looking at emerging technologies to determine how those technologies can be used to enhance our ability to deliver products or increase our value proposition for our customers.

The definitions of Web services vary greatly from person to person. How would you define Web services? What do Web services mean for you and your industry?

Using Web technologies, search, find, and execute business functionality that has been implemented using Web technologies. The service may be a Web site containing content or a program.

The telecommunications industry historically is slow in the adoption of new technologies to support its' business practices. However, the use of Web services as a delivery mechanism for providing services could have a major impact on their operating expenses. Take for example a corporate IT group that has a large network infrastructure. Today you would call your local sales rep to place the order for a new circuit to add capacity. A few days later, you might get confirmation that the circuit has been ordered and be provided with a due date for the installation. Hopefully the sales rep recorded the circuit termination points correctly and the order doesn't stall because of a bad service address. Thirty – sixty days later, you hopefully have a circuit that has been provisioned as you expected.

Now change the model in the world of Web services. From a network design tool, you search the Internet for a company that can provide a circuit from the validated service addresses at your desired price point. Once you find the right provider, you negotiate a contract for the circuit, and start the provisioning process. Status for the circuit is provided to you real-time and the circuit is installed in 10–20 days.

That is the change in the business model that Web services would facilitate for telecommunications. Obviously there are other issues that need to be resolved to completely automate the provisioning process, but at least you are certain that the order is correct.

What do you think about the J2EE vs. .NET as a platform for Web services controversy?

That is a religious discussion. As a software vendor, we look to utilize the technologies that have the widest acceptance in our market, which is telecommunications. J2EE appears to have greater acceptance in the larger carriers than .NET. Part of the religious war gets to their reluctance to use Windows as the base operating system for mission-critical applications such as the ordering of services for customers.

I would look for the common elements of their strategies. Both strategies tout the use of declarative definitions for the interface. XML is the basis for declaration for both strategies. WSDL and UDDI are the implementations backed by both camps.

Can you please outline for us the problem for which you decided to include Web services as a component of your solution? Were there other potential approaches and if so, why did you decide to go for Web services?

Author Bio

Norbert Mikula is industry editor of *Web Services Journal* and has more than 10 years of experience in building and delivering Internet and e-business technologies. He serves as vice chairman of the board of directors of OASIS, is recognized internationally as an expert in Internet and e-business technologies, and speaks regularly at industry events.
NORBERT@SYS-CON.COM

One of our key customers and business partners is our service bureau provider. They use Quintessent's products as part of their OSS Interconnection service bureau. This service allows carriers to match the cost of acquiring a new customer to the revenue without investing their precious capital in software. This service provides a customer with faster time to market, easier entrance to new markets, and the ability to generate revenues more quickly.

Part of our service bureau's original plan was to provide these services via a web based user interface. Most of their initial customers were small carriers who had not purchased software to generate and manage orders. These carriers used the Web interface to enter and track the orders until the service is provisioned.

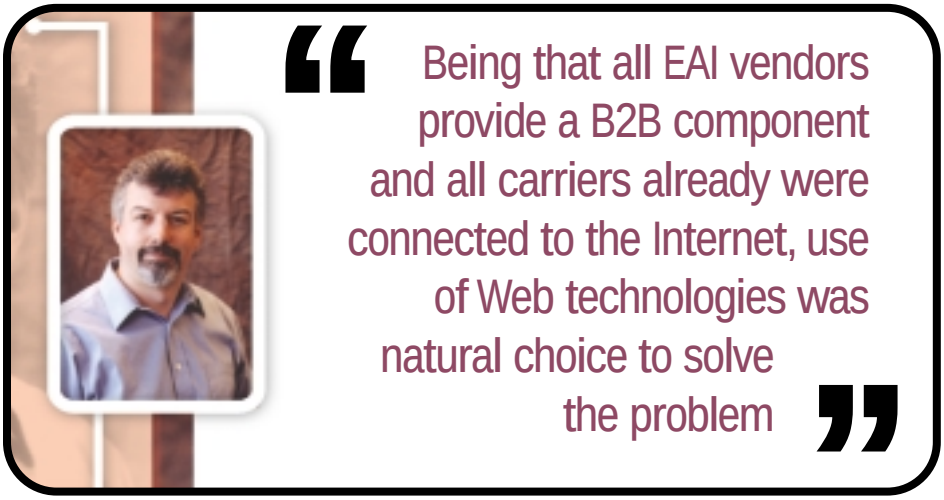
As the telecommunications market suffered through the investment "nuclear meltdown" of 2001, larger carriers who had purchased software to manage services orders but who had not automated the processing of the orders with their trading partners, began to look at a service bureau as a viable option for this function. However, these carriers did not want to enter the order into two different systems and were looking for a solution from our service bureau to provide for the integration of systems in a cost effective manner. As our service bureau software vendor, they turned to us to provide a solution.

As we researched the customer's needs, we found that many of them had selected an Enterprise Application Integration (EAI) product to integrate their back-office computer systems. We also found that a number of them wanted to minimize all operational costs, including circuit charges. Being that all EAI vendors provide a B2B component and all carriers already were connected to the Internet, use of Web technologies was natural choice to solve the problem.

There was one additional issue that required us to look a little deeper at available Web technologies. The ordering of telecommunications services is a lot more complex than filling out a purchase order for office supplies. Many of the products require complex instructions to properly provision the service. Also, not all business scenarios (i.e. new order, change order, cancel order) apply to all types of products that are being ordered. This makes integration of the order management and our software difficult. Developers are required to implement a lot of business logic to determine how to complete the order. We are looking at the implementation of Web services to help solve the problem.

How hard was it to sell the idea of Web services in your company? Who was involved in the decision-making process? Did you encounter any major arguments and if so, how did you address them?

Not really. Quintessent wants to be seen as a leader in our industry. We are always looking at new technologies to determine how they can be used by telecommunication service providers to reduce the time to deliver new services. We formulate an approach and launch a "trial balloon" with our customers, prospects, and business partners. Some trial balloons have floated, while others have not. Use of Web services in our product offering is one that has been getting support.



What were the major problems with respect to Web services that you have encountered during the project?

We are too early in the project to provide a lot of input. I am apprehensive about the potential issues we could encounter when integrating a Microsoft COM environment with a J2EE environment. I think we may also encounter issues depending on the transport and data types we use in the implementation.

I am concerned about the management of the WSDL declarations. The message structures for telecommunication services are very complex. We will be implementing a number of operations and may have multiple bindings per service. Maintaining all of the individual XML files and their relationships could be problematic.

During discussion and at conferences, security seems to be a recurring theme high up on the list of issues for IT experts. What is your view on this? Did you have to address any security-related issues in your project?

Many of the services we will implement require authentication of the user. There are different access levels required to perform certain operations of a service. For example, you would allow a sales rep to pull a customer service record, but you would not allow them to issue the service order. Linking the operation to roles or access levels is important in telecommunications. It would also be very hard to secure or hide the service when you want it broadcast to a wide audience. So, providing authentication and role-based or access—control based security is very important

Web services is still a fairly new and, some would argue, very immature, technology. What would you say? What are, in your mind, missing building blocks of Web services standards and technologies?

In the telecommunications world, trading partner interfaces are frequently changed, usually because of changes in

tariffs for services, new technologies being deployed in the network, and new product offerings. Even though the industry has a standards organization that is tasked to provide standard interfaces for ordering services, the standards body has historically provided implementation specifications for the industry after the trading partners have changed their systems to support the new or changed services. The net result is a variation in the techniques used to support the new or changed services.

All of these changes must be "normalized" by products such as those offered by Quintessent. Changes in the required or optional data elements and their characteristics in the interface have a ripple effect back to the order- management system. The industry needs some mechanism that allows the carrier ordering services to validate that business rules it is using to complete the order are current and if they are not current, what new elements are needed, what has changed, or what has been eliminated. Once the notification of changes is received, tools must be in place that allow a business analyst, not a developer, to correct the data transformation issues created by the changes in the interface.

The authentication and authorization issues must be addressed and linked with approaches already used by IT organizations.

Finally, the use of a configuration repository for maintenance and support of the XML declarations is a must if we are to have a scalable solution for large organizations, such as telecommunications providers.

Have you ever tried to quantify any return on investment (ROI) as the result of your project?

It is too early in the project to qualify this. We will be working with a customer to project the ROI for the project and can report on the success at a later date.

As a last question, what advice would you give anybody out there who wants to jump on the Web services bandwagon?

Be aware that use of Web services is in its infancy. The industry is bound to encounter issues that will require changes in standards and features. I would equate this to the early days of the browser wars. You need to be aware that changes will be required in the interface. However, the availability of components at reasonable costs, helps to reduce that risk.

Pick a small prototype project to get yourself familiar with the technologies and the issues. Work out the kinks in the prototype and then take the outcome of that solution and apply it to something real and tangible.

OVERCOMING Web Services Challenges with Smart Design

Lack of a standard transaction protocol raises issue

In any enterprise computing system, certain operations need to be done as a single atomic unit. Either all of the operations need to succeed or all need to fail in order to keep the system's data internally consistent. All enterprise-class database-management systems support the notion of a transaction, to support exactly this requirement. The transaction coordinator makes sure that all of the database writes (updates or inserts) to a single database, are all either committed or rolled back as a single unit of work. As distributed systems evolved, the Open Group developed the XA Specification to enable multiple transactional systems to cooperate in a single distributed transaction.

An XA-compliant transaction is a two-phase transaction with multiple systems participating. Each of the distributed systems has a resource manager, which controls the transaction state on the single database, message queue, file system, etc. A central transaction manager coordinates the final commit or rollback of all the resources in the transaction. Once each of the resource managers has communicated the ability to commit the transaction, the transaction manager issues a command to all of the systems to commit. If any one of the systems doesn't communicate success in the first phase, then the transaction coordinator would command each of the systems to roll back.

The Java Transaction API (JTA) is the Java implementation of the XA specification; COM+ is Microsoft's XA-compliant transaction manager.

Web services have become a very popular method for connecting distributed systems. The open standards–based techniques that Web services leverage provide many benefits in an enterprise computing environment, including cross-platform interoperability, simple firewall configuration, and limited software deployment requirements.

However, integrating distributed transactions via Web services presents special challenges, especially in relation to interacting with established transaction coordinators. In this article, I'll examine some of the problems, and suggest architectural best practices to avoid or mitigate them.

Challenges with Web Services

Enterprise architects and developers who use Web services to integrate transactional systems need to be aware of the current Web service challenges. Ignoring or not intelligently planning for these issues can lead to poor system performance or corrupt application data, particularly if the Web services are used as an RPC mechanism.

No Transaction-Control Providers Today

Modern distributed systems have come to rely on application servers to act as transaction-control managers for the various resources involved in a transaction. Most databases and messaging products, some file systems, and all transaction coordinators support the Open Group's XA specification for distributed transactions. The ability to coordinate and commit or roll back a transaction across these multiple types of datastores has simplified application development and management.

The Web services model suffers from a lack of any XA-compliant, two-phase commit resource controllers. Several groups have begun work on defining a transaction-control mechanism for Web services, including:

- ***OASIS:*** Business Transaction Protocol
- ***ebXML:*** Business Collaboration Protocol
- ***Tentative Hold Protocol***

However, none of these protocols has yet been finalized and there isn't overwhelming agreement between the various Web services tool vendors on a standard. Web services developers cannot reasonably expect to see a supported production implementation of any one of these standards anytime soon. This means that Web services developers and architects need to carefully consider how they will include Web service transactions among their other database or message queue updates.

HTTP Isn't Reliable

Many problems arise from the basic protocol used to communicate between computers. Most Web services are built on top of HTTP, which is a "best effort" protocol, with no guarantee that a message has reached its destination. In addition, the public Internet isn't a reliable network for any particular packet or message. If a client calls a Web service

Author Bio
David Rader is a partner with Fusion Technologies, a consulting company specializing in application integration and Web services development. Most recently, he's been developing XML-based applications in the small-business portal space.
DAVIDR@FUSIONTECH.COM

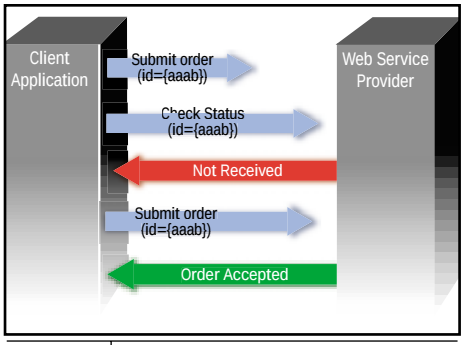


FIGURE 1 **Using unique ID for retries**

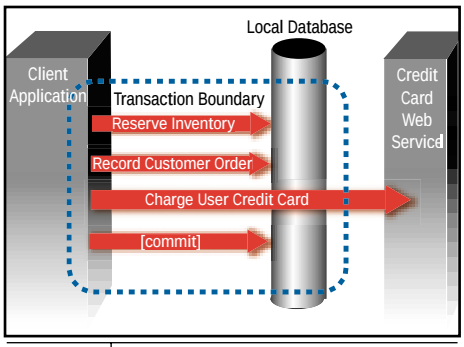


FIGURE 2 **Make Web service call last**

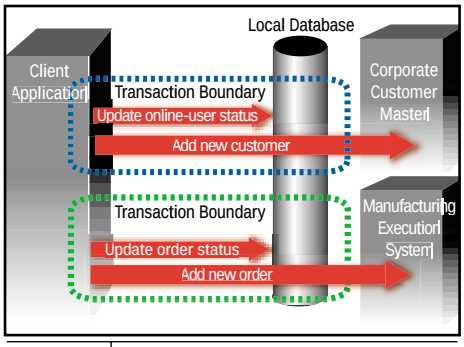


FIGURE 3 **Place Web service calls in separate transaction**

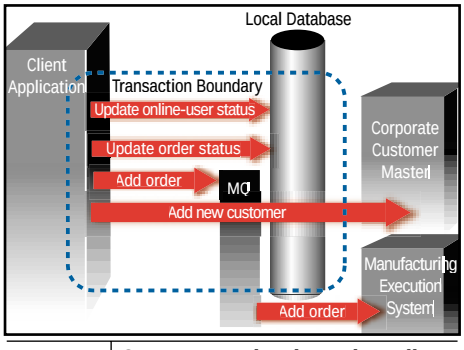


FIGURE 4 **Queue second Web service call**

and doesn't get a response within the timeout period, the client can't be sure of whether the request was received, partially processed, or "lost on the 'Net." Worse yet, if an application retries the request, will the request be duplicated or cause an error? (Two orders entered? Two credit card charges?)

Longer Transaction Time

In general, a Web service "call" will take longer to execute than an equivalent direct query against a local database. The call will be slower due to the HTTP overhead, the XML overhead, and the network overhead to a remote server. In most cases, applications and data providers aren't optimized for XML transport but are translating data from a native format into an XML format. Similar to the effect of distributed object calls, the differences in performance between Web services and traditional database queries need to be factored into the application architecture to prevent unexpectedly poor performance.

Read-Only Is Simple

A Web service that provides read-only access to a datastore can effectively avoid concerns about transactional integrity. This is by far the simplest approach to handling transactional integration (i.e., do nothing!) and can work for many examples of Web services, including public information retrieval services like weather forecasts, enterprise-wide directory or authentication services, read access to the corporate customer master file, and inventory-level queries.

Note that the Web service need only be read-only from the client perspective. The service provider may log information, create audit trails, and update usage databases. The important distinction is that the client isn't relying on the server to record any information. In these situations, if the request doesn't complete successfully, the client application can choose whatever error-handling mechanism is appropriate for the user, including retrying the request, ignoring the error, or querying a different datastore, without fear of duplication or further errors.

Read-Only Doesn't Always Cut It

Not every API can be treated as a read-only request. Most real services will support some level of data updating, for instance, a Web service that supports order entry will need to record the customer order. Let's consider two types of applications that use these services – batch and online – which can generally be distinguished by whether or not a user is waiting for the response of the Web service.

In a *batch-oriented* system, the user isn't actively waiting for the response to the Web service request. In this case, the immediate response time isn't critical, as long as the request is eventually processed. A batch system has some simple options for handling uncertain delivery. The system can simply queue the request and continue to retry until a final success or failure response is received. In fact, the leading Web service vendors, including BEA, webMethods, and Microsoft, all support these features as part of their standard product application server.

On the other hand, applications with *online* transactions are dependent on the response from the Web service for immediate application flow or logic. In these circumstances the user cannot proceed (and accomplish his or her goal) until the Web service has completed processing the request. If the service is known to be unreliable, a custom retry mechanism may be built into the client application to retry a request; however, the application cannot reasonably retry more than once, because the user won't patiently wait, but will abandon the requested action. If the user is a paying customer, your business has just lost revenue! Examples of such time-critical actions include authenticating a user or performing credit card authorization prior to delivery of electronic products.

Handling Retries

Due to the uncertain nature of Web service transactions, it's essential to architect Web service APIs to support a retry or request-status mechanism. This mechanism enables a client application to either retry a request without fear of error, or at a minimum, determine the status of a request to ensure that a request isn't duplicated. The key to supporting retries is to ensure that every request type supports a unique transaction identifier used by both the client and server to identify the request.

Any client that wishes to inquire about the status of a request can use the transaction identifier to query the server. For a

request that was received and processed by the server, the server can respond with the results, or an "in-process" status. If the server hasn't received the request, the client can resend the request safely (see Figure 1). This approach enables the client to avoid duplicate entries and have a measure of confidence that the request was correctly processed.

When implementing the transaction identifier, you must consider several competing priorities. To support robust transactions, the client needs to know the transaction ID before submitting the request. The server must enforce uniqueness across all clients and maintain security and privacy of data. A balance must be achieved in which the client generates the ID, usually based on a standard UUID algorithm. The server validates that the transaction ID is unique, rejecting any duplicate requests that don't originate from the same client. If a single client submits a duplicate request, the server should respond with the same result as for the original request.

No Two-Phase Commit

Until widespread support for a two-phase commit protocol for Web services develops, application architects will plan around the inability to roll back requests. The straightforward approach to handling a nontransactional resource, whether it's a Web service, file system, or other service, is to submit the request to that resource as the last step in an distributed transaction. The outcome of the Web service request will determine if the overall transaction is committed or aborted. This approach can lead to some interesting application logic, in which confirmation records are written to a database before the request is submitted. Since the database transaction can be rolled back, the application has ensured the data will not be in an inconsistent state if the Web service fails. If the application were to wait until after the Web service call to record the confirmation, it may fail to record the result even if the request is successfully processed (see Figure 2).

One requirement of this approach is that you break larger transactions into subtransactions with only one Web service call participating in each subtransaction. Each subtransaction should be semantically correct by itself, with no direct dependency on the following steps. Consider how an order entry system using Web service interfaces for both a corporate customer master and a particular manufacturing execution system (MES) should handle an order for a new customer. If the system logically breaks the overall process into two separate transactions, one to add a new customer to the customer master and a second to place the order, it can avoid any problems with attempting to roll back the first Web service call. If we are unable to connect to the MES to process the order request, the customer has still been logically and completely added to the customer master (see Figure 3).

In a larger system that utilizes functionality provided by many Web services, it may not be possible to isolate different system calls. In this case, it's imperative that the application architecture be carefully planned to minimize the potential for inconsistent data between the participating subsystems.

Queued Processing

In the particular example of submitting to an MES, an alternative approach would be to delay submission. If the MES doesn't apply additional validation rules that cause the order to be rejected, we can assume that order will be accepted. By queuing the order request for later submission to the MES, we can retry as often as necessary to succeed. This approach simplifies the transactional layout, allowing the application to consolidate to only a single logical transaction. However, the application logic may become convoluted, with the customer added to customer master after the order is placed in the message queue for delivery (see Figure 4).

Because queuing systems and databases will both participate in a distributed transaction, we can roll back the queue insert if the customer-add fails. In this way, we can correctly control the data consistency. As an additional side benefit, by delaying delivery to the MES, we also avoid requiring additional 24-hour availability on the existing manufacturing system simply to support the distributed Web service architecture.

Careful Resource Use

The above discussion focused on integrating a Web service update with local transactional resources, and earlier I "dismissed" integrating a read-only Web service as easy. Well, nothing in programming is truly easy, and this is no exception. The longer average call time for a remote system call (like a Web service) can negatively impact the behavior of a transactional monitor. In order to coordinate all the actions within a transaction, the transaction manager needs to lock each resource that has been modified for the lifetime of the transaction. This prevents any other user or process from reading new or changed values until after the transaction has been committed. When a Web service call is executed in the middle of a transaction block, the transaction locks will be held until the Web service completes, because the transaction hasn't yet been committed.

The increase in a single transaction time can potentially have a snowball effect on a high-volume system. By holding locks open for one second longer, this transaction will prevent other transactions that need the same resources from even beginning for an extra second. That delay in start can delay other transactions, and so on. If a system was executing tens or hundreds of transactions per second, adding a single Web service call in the middle of the transaction could wreak havoc with the system throughput. A simple approach to avoiding this is to perform read-only queries before beginning any XA-transactions. The thread that calls the Web service will be blocked while waiting..

Returning to the order entry system, if we need to check the customer's maximum credit in order to enter the order, we can retrieve the maximum credit amount before beginning the order entry. Note that this isn't perfect, in that another system may update the credit amounts after we've read the file. However, that's one of the chances for incorrect data created as a result of the lack of a Web service transaction protocol.

Summary

In this article, I looked at some of the challenges of integrating Web services into transactional systems. Web services provide an easy way to integrate distributed applications via standard protocols, but there isn't yet a standard mechanism for implementing transactional control. Many of the ideas presented here apply equally well to any nontransactional, high-latency external system call. Hopefully, the architectural guidelines and suggestions will help you build more robust distributed systems.

Reviewed by Brian Barbash

AltoWeb Application Platform

Develop, deploy, and manage – without knowing enterprise Java

About the Author:
Brian R. Barbash is a consultant for the Consulting Group of Computer Sciences Corporation. He specializes in application architecture and development, business and technical analysis, and Web design.
BBARBASH@CSC.COM

The AltoWeb Application Platform is a J2EE-compliant system designed to provide an integrated environment to develop, deploy, and manage Web-based and services-based applications. It allows developers to create applications without specific knowledge of enterprise Java and to deploy the final product on J2EE-compliant application servers.

The platform consists of several components: AltoServer, AltoStudio, AltoManager, and AltoMonitor. AltoStudio is the integrated development environment in which applications are built. The AltoServer, running inside a J2EE-compliant application sever, provides the runtime environment and monitoring engine for applications created in AltoStudio. The AltoManager component manages the remote deployment of AltoWeb applications built in AltoStudio. AltoMonitor provides real-time monitoring of application usage and performance. Figure 1, from the AltoWeb Web site, provides an outline of the architecture of the AltoWeb Platform. For this review, I will be focusing on AltoStudio.

AltoWeb is distributed in three configuration packages to run on three individual application servers: IBM's WebSphere, BEA's WebLogic, and the JBoss application server. For this review, I've loaded the WebLogic version of AltoWeb onto WebLogic Server 6.1.

FIGURE 1 **Outline of AltoWeb Platform architecture**

AltoStudio Concepts

AltoStudio is a development environment that follows a component-based approach to building applications. Development is done while connected to a live instance of AltoServer and all objects are deployed automatically as work is saved.

Development work is organized into applications and projects. Applications represent Web application containers on an application server. Projects are collections of components that perform a collection of business processes, such as managing a product catalog. Applications may contain multiple projects.

The developer has at his or her disposal four categories of objects for building applications:

- ***Data Access:*** Provides access to external data stores. Connectors to external sources include SQL databases, Web servers, file systems, XML files, HTML files, and flat files.
- ***Business logic:*** Business logic objects provide the actual logic rules for execution, the mechanisms to link the presentation tier to the data tier, and the means to encapsulate and represent business entities.
- ***Views:*** Views represent the presentation layer of an application and may be created in JSP and XML.
- ***Components:*** Customized, user-developed Java classes that manipulate data.

In addition to the individual editors in AltoStudio for objects such as JSP pages, XML pages, and Java classes, external editors may be registered for each object to give the developer access to the full range of capabilities provided by those specific tools.

Development in AltoStudio

For this review, I'll create a simple application that accesses user information in an Oracle database. The initial setup will involve establishing a SQL Data Source object that will use a connection from an existing pool in WebLogic. AltoStudio provides the ability to access a connection object through a JNDI lookup by simply specifying the name of the connection object. Alternatively, AltoStudio can access a database through a JDBC driver.

The first step to representing the User as an object available to other components is to create a MetaObject. The MetaObject, linked to a specific data store, provides a layer of abstraction isolating the details of the data storage mechanism from the components that will access it. The definition of

AltoWeb

CONTACT:
Steve Wilkes
Principal Technologist
1731 Embarcadero Road
Palo Alto, CA 94303 i
650 251-1591
swilkes@altoweb.com

PRICING:
Platform starts at $3500 per developer seat

TEST ENVIRONMENT:
Pentium III
256 MB RAM
Windows XP Professional
Oracle 8*i*
BEA WebLogic 6.1
JDK 1.3.1

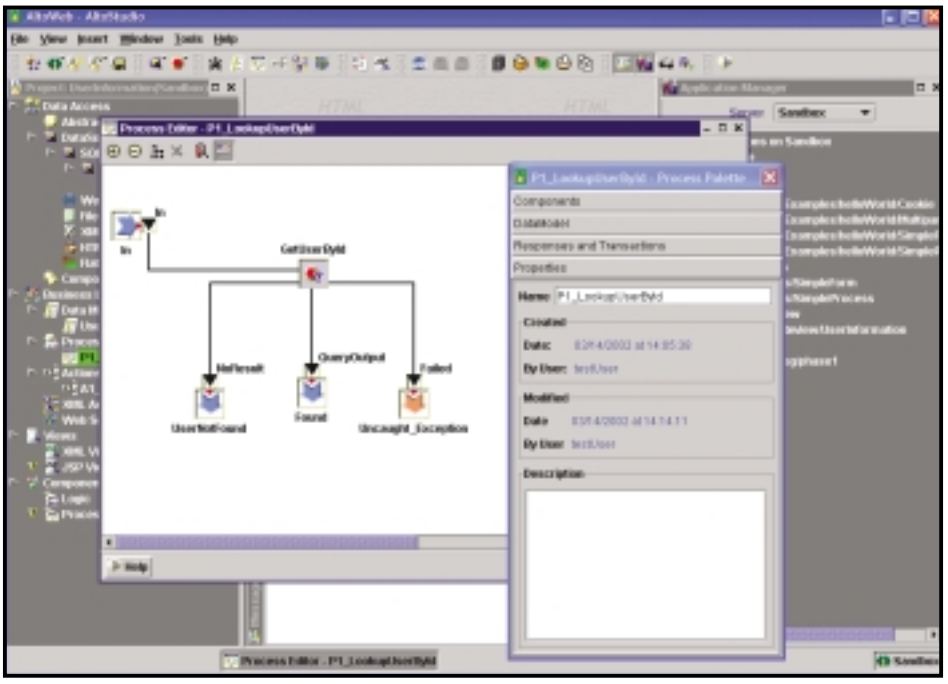


FIGURE 2 **Process diagram**

a MetaObject consists of fields and operations. The fields may be created from a custom SQL statement or selected directly from a list of available tables and columns. Operations are defined to manipulate the item represented by a MetaObject. For example, several Select operations may be defined to handle lookups or searches by various fields. These operations may be created from SQL statements or by applying one of the available filter types.

Once the data sources and the MetaObjects have been created, process logic may be created to perform the business operations on the data. Process objects are created by diagramming the operations in a process view. The process view contains individual component operations, each with their own inputs and outputs.

In the example in Figure 2, the process executes the function GetUserById. The function maps the input parameter to a defined select operation on the User MetaObject and produces three outputs. The function outputs are then mapped to the process outputs.

Information is transported within and among processes by DataModel objects. These containers provide temporary storage for runtime data and may be shared across components.

Along with the provided modules that may be inserted into processes, AltoStudio also provides the ability to create custom components by coding Java classes. When creating a Java class, the developer defines the input, output, and any custom runtime parameters used during execution. AltoStudio then generates the skeleton Java code for a subclass of ProcessWorker. The developer then has the option to launch an IDE that has been associated in AltoStudio to complete development on the class. When finished, this class is available in the Process editor as an executable component.

Once the process has been defined and the appropriate DataModels assigned and created, an interface must be created to allow other processes to act upon it. The interfaces are defined by Action components. Three types of Action components exist in AltoStudio: Actions, XML Actions, and Web Service Actions. The standard Action object is used in browser-based applications to call a target process, custom class or pass data to a public Data Model. The end result of an action is a JSP view or a URL. XML Actions allow AltoWeb to manage client requests in XML format. Web Service Actions allow AltoWeb processes to be executed via SOAP calls. Once defined, the Web Service Actions may be published to a UDDI registry from within AltoStudio.

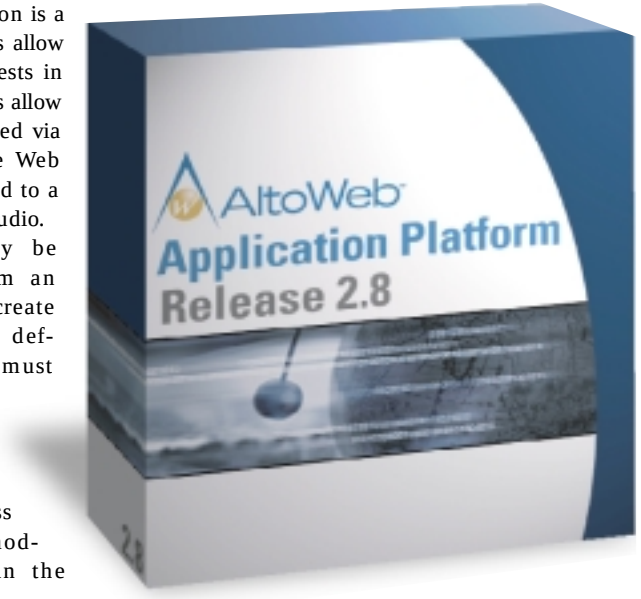
Web Service Actions may be created from scratch or from an existing WSDL definition. To create the action from an existing definition, an active UDDI registry must be available for AltoStudio to browse. Once available, the developer selects the definition desired and AltoStudio creates a new process with the corresponding data models and structures defined in the WSDL document. When creating a service without an existing WSDL definition, the developer must follow specific naming and data structure conventions outlined in the documentation.

Finishing Up

Throughout the development process, AltoStudio is linked with the host application server to store applications. This gives the developer the flexibility to test the application from an external client throughout the development cycle. With the combination of Web Service Actions, JSP Views, and XML Views, the application is available from very distinct types of clients with minimal development effort. The underlying business process is independent of the access method created for that process. In the case of the User Lookup Web Service Action I've created, AltoStudio generated the necessary SOAP call required to test the service and retrieve the results.

Summary

The AltoWeb Application Platform provides an integrated environment to develop, test, deploy, and manage Web-based and services-based applications. The AltoStudio development environment provides developers with the tools to create business-oriented applications without requiring a low-level knowledge of J2EE and Web services technologies. Its flexibility does present a learning curve for new developers, but once acclimated, the tool is effective. Overall, the AltoWeb Application Platform is a solid tool that can improve the development of Web- and services-based applications.



written by Allan Woloshin and Sanjeev Kumar

Taking the LEAP

C++ containers vs C# Collections

While moving from C++ to C# means giving up template-based containers, that doesn't mean you can't effectively organize your data. And like C++, C# collections have some unique benefits.

The concept of computerized arrays has been around almost as long as computers themselves. It allows a program to deal with large quantities of data almost as simply as dealing with a single unit of data. It underlies almost all sorting algorithms. C++, like most other languages, has built-in language support for arrays.

In C++, arrays are always one-dimensional – but you can allocate arrays of arrays to counter that fact. The name of an array is almost always converted into a pointer to its first element, and most array operations work equally well on pointers. For non-rectangular arrays, C++ works equally well with arrays of pointers – allocating and freeing the odd-shaped array can be inconvenient, but otherwise it works extremely well.

Why would anybody want anything more? There are several problems with arrays.

For one thing, you cannot pass an array to a subroutine; instead, C++ automatically converts the array name into a pointer to the first element. This means that arrays are always passed by reference rather than value – just the opposite of most C++ arguments. If the subroutine changes the content of the array, the caller sees the changes, like it or not. This also means that the subroutine doesn't automatically know the size of the array! Many security holes in C and C++ programs can be attributed to functions such as gets(), which accepts the address of a character buffer but not the maximum length.

Converting an array address to a pointer solves many problems, but it causes some, too. C++ has special rules governing the use of pointers, and not all of them are designed to work correctly with pointers. For instance, you can pass an array of Derived to a function that expects an array of Base and the compiler won't complain – but your program won't work correctly.

Arrays are also inflexible. Your program must know the maximum size in advance or else it must be prepared to re-allocate the array (and copy all elements) whenever the array is full. The latter usually means keeping track of two different sizes – the amount of space reserved so far, and the amount of space actually used. In order to "insert" an element into the array, you must be prepared to shift every existing element after that location. Any time you expand an array or insert elements, you also change the address of elements – if another part of the program is a pointer or reference to a moved element, that pointer or reference is now "dangling" and a potential source of major program errors.

Finally, arrays are pure memory devices – unless your operating system has a "virtual memory" capability, every element takes up memory all of the time.

Most of these problems can be solved. You can

Allan Woloshin, a system architect at SEI Information Technology, has become proficient in a wide array of technical languages and tools during his 25 years as a developer and analyst. Allan holds a bachelor's degree in computer information systems, and is a Microsoft Certified Solutions Developer.
AWOLOSHIN@SEI-IT.COM

Sanjeev Kumar, a senior system architect at SEI Information Technology, is an expert in the development of multitier client/server systems for industries including finance, healthcare, and real estate. Sanjeev holds bachelor's and master's degrees in computer science, and is a Microsoft Certified Professional and a Sun Certified Java Programmer.
SKUMAR2@SEI-IT.COM

pass a length parameter along with your array pointer; you can use special naming conventions that make it easy to spot when you've passed the wrong type of array; and so on. The thing is, you have to know what you need. In other words, it's error-prone.

An alternative to the built-in array is the linked list. C and C++ make it easier to build linked lists than many other languages, and this can solve the problem of having to extend the array and of inserting new elements without moving the others. Also, linked lists usually allow mixed types; putting a Derived object in a linked list of Base probably isn't an error. But the list still needs to fit into memory; the size is still unknown unless it's separately maintained; and it's still not possible to pass the linked list by value. Furthermore, linked lists are very error-prone, especially on multithreaded environments or in the presence of exceptions, and linked lists cannot be processed in arbitrary order.

In the late 1980s and throughout the 1990s, Alexander Stepanov and David Musser, and later Meng Lee, developed a library of template classes and algorithms. They had several goals, including algorithms that were fast, powerful, and extensible. The library had many different components, but among the most useful were their container classes. In 1997 they offered their work to the ISO/IEC C++ standards committee, which adopted it, with changes.

C++ Containers

The C++ language standard (ISO/IEC 14882-1998[E]) says in Clause 23 that containers are "components that C++ programs may use to organize collections of information," and adds: "Containers are objects that store other objects. They control allocation and deallocation of these objects through constructors, destructors, insert and erase operations." It goes on to list the two general types of containers: sequences and associative containers.

The C++ standard says, "A sequence is a kind of container that organizes a finite set of objects, all of the same type, into a strictly linear arrangement." The three types of sequences are *vector*, *list*, and *deque*.

- ***vector:*** The closest approximation to a built-in array, it avoids most of the problems listed above. A vector maintains both a size and a capacity. vector<bool> is specialized to minimize space by packing true/false values in memory, at the cost of speed and code bloat.
- ***list:*** The closest approximation to the linked list I spoke of above. It's much slower for random access, but it allows items to be inserted quickly at any point
- ***deque:*** A compromise between the two. Like a vector, it's quick for random access. It can also insert elements at either end quickly, although inserting elements in the middle isn't as quick as with list.

When you iterate through all the elements of a sequence, you get them "in order" – the first element comes first.

The C++ standard says associative containers "provide an ability for fast retrieval of data based on keys." Practically speaking, this means that the index doesn't have to take an integer. (If it does take an integer, the values don't have to be contiguous; this is known as a "sparse array.") The four types of associative containers are *map*, *multimap*, *set*, and *multiset*.

- ***map:*** Holds data that can be accessed by a key. For instance, a map of customers might be accessed by customer number.
- ***multimap:*** Much the same thing as a map, except that it allows duplicate keys. For instance, two library books might have the same Dewey Decimal number.
- ***set:*** Much like a map, except that the data is the key. A good example is an English-language dictionary; if the word is in the set, it is properly spelled.
- ***multiset:*** Much the same thing as a set, except that data can be in the multiset more than once.

The associative containers can only contain elements that can be sorted – you can either make sure that operator less-than works correctly, or else you can construct a class object that knows how to sort the contained objects. When you iterate through all the elements of an associative container, you get them in that sorted order.

The library also defines *container adapters* such as *queue*, which accepts a sequence such as list or deque, and ensures that access is always at the beginning or end; *stack*, which does much the same thing, except that elements are popped off in reverse order from when they were pushed on; and *priority_queue*, which returns the lowest element first.

These standardized containers solved a lot of problems. With such a rich assortment to choose from, there is bound to be one that satisfies performance requirements for almost any need. Standard C++ library routines understand how to access data in these containers, and there is a rich variety of algorithms included in the library – algorithms for copying, sorting, even I/O.

Unfortunately, there were several problems that standardized C++ containers simply couldn't solve. For instance, the standard containers require all elements to be in memory at once. (However, they provide a framework you can use to develop your own container without this requirement – and your container will work with all Standard C++ algorithms.) Mixed data types are still a major issue.

Also, it's still a bad idea to have a reference or pointer to a contained element, except under certain very well-defined circumstances. To make it possible to iterate through all of the elements in a container, we need some other mechanism. These are called *iterators*.

C++ Containers Cannot Contain Mixed Types!

We hinted above that C++ containers have a problem with mixed data types, such as a hierarchy of classes. We've written some demonstration code in Listing 1 (Listings 1–4 are available on the Web at www.sys-con.com/webservices/sourcec.cfm). The programmer wanted to display employee information for Wilma, and then the customer information for Barney. But that probably isn't what he'll get. people is a vector of person, so you can't put anything into people except objects convertible to person. employee and customer are convertible to person, so the code above compiles without error or warning messages. Actually, the person subobject is copied into the vector. In other words, the integer i is copied, but not the cname or ename.

The same thing happens with arrays:

```
void bar() {
 Person people[5];
 // Copies the person subobject of
Wilma
 People[0] = Wilma;
 // Copies the person subobject of
Barney
 People[1] = Barney;
}
```

If you must store hierarchical items in an array or container, you do so by creating a container (or array) of pointers to person, as we've shown in Listing 2. The last part of the code sample illustrates a problem with this technique. You must delete all of the elements whenever the vector is going to be deleted, or else you get memory leaks. This is both tedious and error-prone.

How C++ Iterators Work

In C++, an iterator is a generalization of a pointer. In fact, any standard library algorithm that accepts an iterator will also accept pointers to an in-memory array. But iterators can do much more than simple pointers do.

The C++ standard defines five different types of iterators:

- ***Input iterator:*** Allows you to read data. In other words, you can dereference it on the right side of an assignment expression. You must increment an input iterator once (and only once) between each read.
- ***Output iterator:*** Allows you to write data. In other words, you can dereference it on the left side of an assignment expression. You must increment an output iterator once (and only once) between each write.
- ***Forward iterator:*** Allows you to read and write data and increment the pointer. You can move arbitrary distances using the algorithm std::advance(), but be aware that the time taken might be proportional to the distance you move. For instance, std::advance(I,5) might take five times as long as std::advance(I,1).
- ***Bidirectional iterator:*** A forward iterator that also allows you to decrement the pointer.
- ***Random iterator:*** A bidirectional iterator that also allows you to move arbitrary distances in constant time. For instance, pointers are random iterators. Besides algorithms such as std::advance(), you can also use the index operator[], and you can do the same type of math operations that you can with pointers. (For instance, you can subtract two random iterators to find the distance between them.)

All C++ containers have methods (member functions) that return iterators that point into the container. The type of iterator returned depends on the container type.

- deque and vector have random iterators.
- list has bidirectional iterators, as do map, multimap, set, and multiset.
- Nothing in the standard library creates forward iterators that aren't also bidirectional iterators; however, if you create one yourself, it will work with many different standard algorithms.
- The library defines input iterators that work with an istream, and output iterators that work with an ostream.

Most algorithms take not one, but two iterators to define a "range" of data. One iterator marks the beginning of the data and another marks the position *past* the end. This arrangement seems strange at first, but it turns out to be rather convenient. If the iterators are bidirectional or random, you can determine the number of included elements by subtracting the two. There is no need to use special flags to define an empty range; simply pass the same iterator value for beginning and past-the-end. Also, it makes testing for the end of a range simple; with some iterator types, you cannot use less-than or greater-than to tell if the range is finished, but a special sentinel value can be used to mark the end, and when input or output is finished, this value is returned.

When manipulating a container, it's important to consider its effect on existing iterators. For instance, if you add an element to a vector and the internal capacity has to increase, iterators, except for end(), are not necessarily valid any more. This is especially inconvenient when you try to examine and manipulate a container at the same time; for instance, if you write a program that erases entries from a deque of customers, the act of deleting a customer might make you "lose your place." Many operations return new iterators; for instance, erase() might return an iterator to the element after the one that was deleted. Other operations are more problematic.

There is a lot more to C++ iterators; we've only scratched the surface in this article. Consult a good C++ textbook to find out more.

List and Explain C# Collections

Perhaps the most useful C# collection isn't considered a collection at all. C# arrays are much more powerful than C++ arrays, and they natively support many operations that require a class in C++. For instance, the Length property tells you the number of elements in the array, like the size() method in C++ containers.

C# arrays can be made multidimensional in the same way that C++ arrays are: by creating an array of arrays.

```
int[][] jagged = new int[3][];
jagged[0] = new int[3]; // 3 elements
jagged[1] = new int[8]; // 8 more
jagged[2] = new int[4]; // 4 more,
total 15
int q = jagged[1][4];
```

They can also be multidimensional in the classic sense, although of course this must be rectangular.

```
// Allocate 20 elements.
int[] normal = new int[4,5];
int r = normal[1,4];
```

However, C# arrays don't support dynamic resizing or inserting in the middle, any more than C++ arrays do. If you want to do this type of operation, you need to use one of the collection classes. The list of standard C# collection types is even more rich than the list of standard C++ container classes.

- ***ArrayList:*** A dynamically sized array that can contain any sort of object
- ***BitArray:*** A compact array of bit values whose value can be either true or false
- ***HashTable:*** Maintains a sorted list of key/value pairs that can be accessed by key value
- ***Queue:*** Represents a first-in, first-out collection of objects
- ***SortedList:*** A sorted list of key/value pairs that can be accessed by either key or index
- ***Stack:*** Represents a last-in, first-out list of objects
- ***StringCollection:*** Maintains a collection of strings

C# Collections Are Weakly Typed

Unlike C++ containers, C# collections can contain any type of object. You can easily mix employees, customers, and even nonperson objects in any of the C# collections without worrying about memory corruption or unintentional object splicing.

This eliminates an entire classification of hard-to-explain, hard-to-find errors. But this freedom doesn't come for free. C# collections return the contained object as a pointer to an object; that means most of your custom methods won't work correctly until you fix the pointer type.

```
CollectionOfFoo[j].bar(j); // Won't
    compile
((Foo)CollectionOfFoo[j]).bar(j); //
```

TABLE 1: Comparison of C# Collections & C++ Containers

C#	C++
ArrayList	vector
BitArray	bitset
HashTable	map
Queue	queue
SortedList	(No good equivalent)
Stack	stack
StringCollection	vector<string>

```
Okay
```

At the root of this minor inconvenience is a more devilish problem. Even though you know that your collection holds only Foo objects, the compiler doesn't know this. Nothing prevents you from inserting an integer into a collection of Foo objects. Every time you access one of the collected objects, the program silently renegotiates the object type. This means that using C# collections will be at least slightly slower than equivalent C++ containers, although the difference will probably be quite minor.

Comparison of C++ Container Classes with C# Collections

There are some obvious parallels between C++ containers and C# collections. Although a C++ queue<> isn't the same as a C# queue, the two are equivalent enough that they can usually serve the same purpose. Table 1 shows the comparison.

C# does support the equivalent of a C++ forward iterator, but this is rarely used. Instead, elements in C# collections are generally accessed in one of two ways: through the indexer or through a *foreach* statement.

To use the indexer, simply use familiar array notation to access an element. Like C++, different C# collections can use different data types for the index. For instance, a HashTable might let you look up a customer by Customer Number.

If you wish to process an entire collection or search through the elements in order, you should use the foreach statement.

How *foreach* Works

One feature that sometimes makes C++ programmers jealous of their VB counterparts is the for each loop.

Consider a simple for loop and array access in C++:

```
for(int j = 0; j <
mylist.Count(); j++)
    {
 dosomething(mylist[j]);
}
```

The programmer that writes this has to pay careful attention to a variety of minor issues.

- ***Initialization:*** If j is not initialized properly, the loop may not execute at all.
- ***Termination:*** If the Boolean expression (j < list.Count()) is not correct, the loop might terminate early – or not at all.
- ***Incrementing:*** With integers it won't matter, but sometimes the difference between preincrement and postincrement is phenomenal!
- ***Standards:*** Is count() the correct way to get the number of elements, or should it be size() or getcount() or something else?

With the introduction of the foreach loop in C#, problems like these are outdated. foreach iterates through each element in a collection. A simple foreach loop may look like:

```
int[] numbers = {1,2,3,4};
foreach (int j in numbers) {
 dosomething(j);
}
```

You must have already noticed the ease of use of foreach loop and how it took care of lot of potential problem areas in the for loop. With the foreach loop, there is no special setup required for the loop, no special exit condition, no need for a function to get the size of the collection, and no need to fetch individual items in the collection. Deciding how many times to loop is the responsibility of collection, relieving the programmer of this burden.

A foreach loop allows iteration for any class implementing the IEnumerable interface. This includes all C# collections and collection classes included in namespace System.Collections. You can also write your own collection class and make it work with foreach by following IEnumerable guidelines.

foreach will automatically place the fetched element into a temporary variable. Note that the temporary variable j cannot be modified, i.e., j is read-only. Attempts to modify j will produce a compile error.

```
foreach(int j in numbers) {
 j++; //will fail to compile.
 dosomething(j);
}
```

How to Write Your Own C# Collection

Our first advice is: Don't do it if you don't need to! The collections included with C# are quite comprehensive, and handle a wide variety of different uses. Not every conceivable use, of course. If you have some strange memory constraints, or some need to use more than one key at the same time, you might have good reason to write your own collection.

The easiest way to create your own collection is to implement the indexer property. A simple indexer property might look like this:

```
public object this[int index] {
 get {
    if (index > -1 && index <
this.List.Count) {
       return (this.List [index]);
    }
  }

  set {
    if (index > -1 && index <
this.List.Count) {
       this.List [index] = value;
    }
  }
}
```

The index does not have to be an int; it can be any data type you wish to support. You use this value to access the correct element. A get access occurs when your indexer is used anywhere in an expression except on the left-hand side of an assignment.

```
Foo f = myCollection[3];
```

C# automatically calls your property get code with index set to 3. The set access occurs, naturally enough, when the index expression is on the left-hand-side of an assignment statement.

```
myCollection[3] = f;
```

In Listing 3, we implement an indexer and

access objects using it.

What happens when client code tries to use a foreach loop instead of using an indexer?

```
class MyClass {
  static void Main(string[] args) {
    MyCollection words = new
MyCollection();
     foreach(string str in words) {
      Console.WriteLine(str);
    }
  }
}
```

This code will fail to compile, sending a message that it cannot find GetEnumerator(), which is a method exposed by the IEnumerable interface. For a class to be usable in a foreach loop, it must support the IEnumerable interface, which is defined in System.Collections.

Perhaps surprisingly, IEnumerable has only one method; IEnumerator GetEnumerator(). GetEnumerator() returns an instance of interface IEnumerator, which is also defined in the System.Collections namespace. The IEnumerator allows unidirectional iteration (read-only access) of items in a collection. IEnumerator publishes two methods: bool MoveNext() and void Reset(); and one property, object Current {get;}.

MoveNext(), like the name suggests, moves the enumerator to the next element in the collection. It returns true if the operation was successful, or false: if there aren't any elements left to enumerate.

Reset() sets the enumerator to the initial position, which prepares the Enumerator for iteration over the collection. *Note:* Initial position is not the first element in the collection; rather, it's a virtual position before the first element in the collection.

Current {get;} gets the current element in the collection. If the last call to MoveNext() returned false, or if MoveNext() hasn't even been called after Reset(), then Current {get;} throws an IndexOutOfRangeException.

To make our class support all of the same types of access that other C# collections do, all we need to do is derive our class from IEnumerable, define a support class from IEnumerator, and then implement GetEnumerator() by returning an instance of our new class. (*Note:* It's possible to have our class derive from both IEnumerator and IEnumerable, and GetEnumerator() would just return this. We don't recommend this, because it isn't thread-safe.)

Listing 4 shows all the changes required to implement an enumerator for use with the foreach construct. This class can be used the same way as any other C# collection class.

Conclusion

C# doesn't have templates or the STL, so it can't use C++ container classes. C# does have a rich assortment of collection classes. In C++ it's usually dangerous to mix types within one container; C# always allows this, but there are disadvantages to this approach. The C# for each statement is a convenient way to process all elements in a collection. It's surprisingly easy to write your own C# collections that work just like the library versions.

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

THE FIRST & ONLY WEB SERVICES RESOURCE CD!

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES

EDITED BY SEAN RHODY

$119 CD VALUE FROM WEB SERVICES JOURNAL

web services EDGE conference & expo

$100 coupon Inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published since the first issue

- XML in Transit
- XML B2B
- Java & XML
- The XML Files
- XML & WML
- Voice XML
- SYS-CON Radio
- XML & XSLT
- XML & XSL
- XML & XHTML
- 2B or Not 2B
- XML Script
- XML Industry Insider
- <e-BizML>
- XML & Business
- XML Demystified
- XML & E-Commerce
- XML Middleware
- XML Modeling
- CORBA & XML
- Data Transition
- XML @ Work
- XML & Databases
- Electronic Data Interchange
- Ubiquitous Computing
- Information Management
- Objects & XML
- XML Pros & Cons
- Java Servlets
- XML Filter
- UML
- Integration
- WSDL
- Beginning Web Services
- Web Services Tips & Techniques
- Frameworks
- Management
- Security
- UDDI
- .NET

3 YEARS
25 ISSUES
400 ARTICLES
ONE CD

Special Limited Time Price

Now Shipping

$79 + S&H

ONLINE ORDER AT JDJSTORE.COM

SAVE $40

www.JDJSTORE.com

OFFER EXPIRES JUNE 30, 2002

WSJ News

IBM, Microsoft, and VeriSign Announce New Security Spec

(Redmond, WA; Armonk, NY; Mountainview, CA) – Microsoft Corp., IBM Corp., and VeriSign Inc. have announced the publication of a new Web services security specification to help organizations build secure, broadly interoperable Web services applications. The three companies jointly developed the new specification, known as WS-Security, and plan to submit it to a standards body.

WS-Security is the foundation for a broader road map and additional set of proposed Web services security capabilities outlined by IBM and Microsoft to tackle the growing need for consistent support of more secure Web services. The proposed road map, titled "Security in a Web Services World" and authored by Microsoft and IBM, outlines additional Web services security specifications the companies plan to develop along with key customers, industry partners, and standards organizations.

www.ibm.com, www.microsoft.com, www.verisign.com

Microsoft Extends XML Web Services Support In .NET Enterprise Server

(New Orleans) – Microsoft Corp. has announced the Exchange 2000 Server XML Web Services Toolkit for Microsoft .NET, and beta releases of SQL Server 2000 Notification Services and SQL Server 2000 Windows CE Edition (SQL Server CE) version 2.0 to further enable developers to push XML Web services into every corner of the .NET Platform.

The Exchange 2000 Server XML Web Services Toolkit for Microsoft .NET provides the tools and resources developers need to bring contextual collaboration to .NET-based applications using Exchange 2000. Visual Studio.NET developers can tap the services of Exchange, such as calendaring and scheduling, contacts, workflow, and messaging to add to the context of a larger solution. The toolkit provides sample code, white papers, how-to videos, and a self-paced training course to accelerate the design and development of XML Web services.

www.microsoft.com

Actional Accelerates Enterprise Web Services Adoption with SOAPswitch

(New Orleans) – Actional , a provider of cross-platform Web services infrastructure solutions, has announced general availability of SOAPswitch, a Web services gateway that allows a company to successfully publish, secure, and manage Web services. It transforms existing software assets into Web services and is the only such solution to combine extremely rapid integration with centralized management and security of Web services.

SOAPswitch incorporates native understanding of a broad set of applications, providing built-in connectivity to existing software assets. It includes support for packaged applications such as SAP, Peoplesoft, and Siebel, as well as custom applications based on Java, CORBA, COM, or MQ series and is a standards-based Web services framework that complies with the latest Web services standards, including SOAP, WSDL, and UDDI.

www.actional.com

Altova Releases XML Tools Integrated With Visual Studio .NET.

(New Orleans) – Altova Inc, the provider of XML Spy 4 Suite, has announced their membership in the Microsoft Visual Studio .NET Integration Program (VSIP) and the immediate availability of an XML Spy for Visual Studio .NET Integration Kit which will enhance developer productivity in building XML-based Web services for the Microsoft .NET platform.

The availability of XML Spy's XML developer environment directly within the Microsoft Visual Studio .NET Integrated Development Environment (IDE) integrates XML Spy's advanced XML editing capabilities with the development environment of Visual Studio .NET.

XML Spy for Microsoft Visual Studio .NET Integration Kit is available immediately for download at www.xmlspy.com.

www.altova.com

Jacada Announces EJB "Wrapper" for Legacy Business Logic

(Atlanta) – Jacada Ltd. has announced that Jacada Integrator now provides native support for J2EE. Jacada Integrator is deployable as an EJB and can run natively on any application server that supports J2EE development, including IBM's WebSphere Application Server and BEA WebLogic Server. By utilizing Jacada Integrator as an EJB in these environments, developers can easily integrate existing legacy business logic and data into newly developed enterprise systems.

www.jacada.com

Grand Central Communications Unveils New Release of Its Web Services Network

(San Francisco) – Grand Central Communications, Inc. , provider of a leading Web services network, has announced that it has enhanced its network-based service to enable its customers to more quickly add security, reliability, and monitoring to their inter-enterprise Web services deployments. These enhancements leverage WSDL (Web Services Description Language) to transparently provide advanced integration capabilities when using any of the major Web services development platforms, including those from BEA, IBM, and Microsoft. New functionality includes WSDL Messaging, enhanced Verisign security, and exception alerts.

www.grandcentral.com

A recent online poll at www.sys-con.com/webservices asked the question, "Will you be using Web services to create UDDI business service descriptions?" The majority of the responses, shown below, indicate a definite interest in Web services:

31% A. Within 3 months

47% B. Within 6 months

11% C. Next year sometime

6% D. Never

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Actional	www.actional.com/soap2	800-808-2271	6
Altova	www.altova.com		19
Dynamic Buyer	www.ibm.com/smallbusiness/dynamicbuyer	800-426-7235 ext.5014	43
engindata Research	www.engindata.com		61
eXcelon	www.exln.com	800-962-9620	17
Hewlett-Packard Company	www.hpmiddleware.com/download	856-638-6000	67
iWay Software	www.iwaysoftware.com/guaranteed	866-297-4929	11
Java Developer's Journal	www.sys-con.com	800-513-7111	31
JDJ Edge Conference & Edge	www.sys-con.com	201-802-3069	37
JDJ Store	www.jdjstore.com	800-303-JAVA	62, 63
New Atlanta Communications	www.newatlanta.com		15
Simplex Knowledge Company	www.skc.com/training	800-620-3700	47
Sonic Software	www.sonicsoftware.com		2, 3
SpiritSoft	www.spiritsoft.com/climber		4
SYS-CON Media	www.sys-con.com	800-513-7111	39, 57
Web Services Edge Conference & Expo	www.sys-con.com	201-802-3069	33, 34, 35
Web Services Journal	www.sys-con.com	800-513-7111	53
WebLogic Developer's Journal	www.sys-con.com	800-513-7111	49
WebSphere Developer's Journal	www.sys-con.com	800-513-7111	25
Wireless Business & Technology	www.sys-con.com	800-513-7111	41
XML Edge Conference & Expo	www.sys-con.com	201-802-3069	29
XML-Journal	www.sys-con.com	800-513-7111	45
Sitraka	www.sitraka.com/jclass/ws		21

WS-I & W3C

Anne Thomas Manes

Anne Thomas Manes is the CTO of Systinet, a Web services infrastructure company. Anne is a recognized industry spokesperson and has published on a range of technology issues.
ATM@SYSTINET.COM

IBM and Microsoft recently launched another Web services-related effort – the Web Services Interoperability Organization (www.ws-i.org), or WS-I for short. Its charter is to promote Web services interoperability across platforms, operating systems, and programming languages. I, for one, view interoperability as absolutely critical to the success of Web services technology. I don't think I'm alone in the view, since more than 60 companies joined the consortium within the first week. Obviously the formation of this group is a "good thing."

So I was a bit amused by the immediate defensive response that occurred after the announcement. How does WS-I relate to the World Wide Web Consortium (W3C – www.w3.org)? Isn't W3C supposed to define the interoperability standards? And what about the SOAPBuilders group (http://groups.yahoo .com/group/soapbuilders)? Aren't they taking responsibility for SOAP interoperability?

Interoperability has always been a core goal of all the various Web services platform vendors, and an informal organization centered around the SOAP Builders discussion group has been working for the past year on this issue. This group holds quarterly interoperability labs at which the vendors get together, test their products, and uncover interoperability issues. The last interoperability lab (Round 3) was sponsored by IONA Technologies in February 2002. (www.whitemesa.net/r3/interop3.html).

I don't expect WS-I to compete with the SOAPBuilders group. The SOAPBuilders participants work at uncovering and fixing interoperability issues. The WS-I organization will define profiles that help users easily understand what levels of interoperability they can expect from various Web services technologies. WS-I will publish white papers, guidelines, and other documentation to guide developers. They also plan to develop tools and test suites to help developers and vendors verify that their Web services comply with the standards. Obviously the two efforts are complementary.

The relationship between W3C and WS-I should also be complementary, although for the moment their efforts are orthogonal. W3C is working on building the next generation of Web services standards. WS-I focuses on ensuring interoperability using today's Web services technology: SOAP 1.1 (www.w3.org/TR/SOAP), WSDL 1.1 (www.w 3.org/TR/wsdl), and UDDI 1.0 (www.uddi.org/specification.html). None of these specifications are W3C standards.

"Wait," you say. "What do you mean SOAP and WSDL aren't W3C standards? Look at the URLs. You get them from W3C. They must be W3C standards." While it's true that W3C has posted these specs, they're not official W3C Recommendations. These specifications (and the intellectual property contained within them) are, in fact, owned by IBM, Microsoft, and a few of their closest friends. If you look on the first page of both the SOAP and WSDL specifications, you'll find this little disclaimer:

This document is a NOTE made available by the W3C for discussion only. Publication of this Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members. W3C has had no editorial control over the preparation of this Note. This document is a work in progress and may be updated, replaced, or rendered obsolete by other documents at any time.

As I said, W3C is developing the next generation of SOAP and WSDL. The current SOAP and WSDL specifications aren't perfect. They contain errors, omissions, and inconsistencies. It's appropriate to have a formal venue in which to nurture and develop these specifications. W3C initiated the XML Protocol Working Group (XMLP – www. w3.org/2000/ xp/Group) in September 2000 to develop a standard XML Protocol based on the SOAP 1.1 specification. Draft specifications of W3C SOAP 1.2 are available on the XMLP Web page. W3C SOAP is now described by two specifications:

- ***Part 1: Messaging Framework*** (www.w3.org/TR/soap1 2-part1): Defines the basic SOAP envelope and the protocol binding framework.
- ***Part 2: Adjuncts*** (www.w3.org/TR/soap12-part2): Defines SOAP encoding, SOAP RPC, features and binding conventions, message exchange patterns, and the HTTP binding.

The W3C Web Services Description Working Group (WS-Desc; www.w3.org/2002/ws/desc) was formed in February 2002, so they're just getting started. It will be a little while before they manage to produce a specification.

Both the XMLP and WS-Desc groups are part of the new W3C Web Services Activity (www.w3.org/2002/ws). There's one other working group in this activity, the W3C Web Services Architecture Working Group (WS-Arch; www.w3. org/2002/ ws/arch). The goal of the WS-Arch group is to produce an architecture that supports a fully automated, functional, distributed system based on Web services. The architecture must identify the technologies necessary for Web services to be described, discovered, and used. The WS-Arch group must also identify gaps in the architecture and propose new technologies that must be developed to fill those gaps.

As Web services technologies mature and people start building serious application systems requiring security, reliability, and transactions, these gaps will become much more apparent. For these applications, interoperability will require more than just guidelines and white papers. It will require new standard specifications that allow heterogeneous Web services to exchange security information and transaction context.

Today, Web services platform vendors are starting to add advanced features to their products. Unfortunately, there's no consensus among the vendors on how to represent information on things such as remote references, sessionIDs, or security context, and each vendor does so in a slightly different way. These little details make interoperability challenging. Without standard specifications, the only way to guarantee interoperability is to take a least-common-denominator approach and to avoid using any of these advanced features. WS-I will help define the least common denominator. W3C will define the standards that allow us to take Web services to the next level.