Sun Open Net Environment (Sun ONE)

by: Dean Calabrese

Copyright 2003, Faulkner Information Services. All Rights Reserved.

Docid: 00017997

Publication Date: 0304

Publication Type: TUTORIAL

Preview

Sun Open Net Environment (Sun ONE) is Sun's one-stop-shopping, Java-centric, Services on Demand solution that promises an Internet software infrastructure capable of providing data and applications to anyone, anytime, anywhere, on anything. It competes directly with Microsoft's .NET application framework, and encompasses everything from architecture to standards to enterprise system implementations. In addition to hosting Web services, Sun ONE is compatible with ebXML and various other industry-based standards. It uses the DART (Data, Applications, Reports, and Transactions) approach to creating and delivering services on demand. It is Sun's attempt to provide a comprehensive design guide, software platform, and application development language.

Report Contents:

- Executive Summary
- Issues to Consider
- Analysis
- Recommendations
- Web Links

Executive Summary

[return to top of this report]

Sun ONE is an all-inclusive platform for developing and deploying B2B (Business to Business) or A2A (Application to Application) applications.

It integrates back-end applications through compliance with industry standards such as SOAP, J2EE, WSDL, UDDI, various flavors of XML, XSLT, LDAP, and Java MS.

The Sun ONE platform allows developers to create, deploy, and manage business solutions that integrate new and legacy systems, data, and applications into automated process flows. These solutions or services may reside on the Internet as Web services, or on local or wide-area networks, and may include both legacy and new technologies.

Sun ONE's process engine connects the elements that control and manage the entire business process, thus enabling the development of flexible enterprise applications that can convert existing components into powerful, graphically depicted process definitions. The message transport is a two-way conduit for...
XML/SOAP-based data messages sent over HTTP/HTTPS or Java MS (Java Message Service).

From a business perspective, the benefits include:

- A unified view of community, transactions, and infrastructure.
- Consistent and clear tracking of IT progress and issues.
- Re-use of components, applications, and services for Intranet and extranet construction.
- Anytime, anywhere, anyone, any device services for communities.

Sun ONE’s Integration Server EAI Edition, a component of the Sun ONE Integration Family, transforms data by means of an XSL processor that utilizes XSLT (a Web standard) to transform XML documents. This translates data elements among multiple dialects of XML. The Sun ONE Integration Family offers technology adapters for common component technologies, custom, mainframe, and packaged systems, and it contains component support for Java, COM/ActiveX, CORBA, and C++. The Sun ONE Connector Builder facilitates the rapid development of robust J2EE Connector Architecture (JCA)-based custom adapters.

The Sun ONE Integration Server B2B Edition, another component of the Sun ONE Integration Family, is an Internet commerce exchange application that enables an enterprise to automate and manage business processes between organizations over the Internet and existing private networks. It enables encrypted transmission of documents and messages among heterogeneous trading partner systems, and it can transform information from ONE format into another by providing the ability to define partners by transaction workflow-based requirements, rather than by technology-based requirements.

Some of the benefits to programmers include:

- Use of a built-in version of Tomcat that can be managed easily from within the IDE is included, making packaging of Web applications into WAR files, developing custom tag libraries, debugging Web applications, and HTTP transaction monitoring extremely easy.
- The indentation engine can be configured easily to match the developer’s preferred indentation mode. Auto Comment, which automatically generates documentation, is another useful feature. It generates proper Javadocs, checks existing ones, and auto-fixes most Javadoc errors.
- With Sun ONE, it takes only a few clicks to create a decent GUI. With right-clicking, users can add event handling as well.
- Integration with Apache Ant (a de facto standard for Java tools) is included, along with some useful Help files.
- Additional features include a built-in CVS client, an image viewer, an auto-complete option, and a powerful XML display and edit capability.

Several products that have been renamed to emphasize Sun ONE products are optimized to work best with other members of their own “family.” Sun emphasizes that Sun ONE is based on open standards, so existing systems may easily interoperate with it. The implication, of course, is that this interoperability is a stopgap for users to employ until they fully implement Sun ONE. Table 1 lists Sun Microsystems' suite of Sun ONE (formerly iPlanet) products.

<table>
<thead>
<tr>
<th>Current Name</th>
<th>Former Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun ONE Web Server</td>
<td>iPlanet Web Server</td>
</tr>
<tr>
<td>Sun ONE Portal Server</td>
<td>iPlanet Portal Server</td>
</tr>
<tr>
<td>Sun ONE Application Server</td>
<td>iPlanet Application Server</td>
</tr>
</tbody>
</table>
The Solaris operating system and the STAR software suite are also included in the Sun ONE platform.

**Issues to Consider**

The big question is still surround Web services, and Sun's extension, Services-on-Demand. Major questions still exist that have nothing to do with the platforms, and everything to do with the philosophy. How will companies determine whether Web-based Services or components are trustworthy, and do what they say they do? How will reliability, scalability, and security--all significant concerns in these turbulent times--be addressed? Who will maintain applications constructed from components from diverse sources? These questions need to be asked about any Web service platform, not just Sun ONE.

Sun ONE provides loose integration, without the cost of an extensive middleware integration effort. It is very effective at enabling access to previously inaccessible data, but less so at providing the reliable unification of data repositories.

Projects are best suited for a Services-On-Demand approach when existing applications retain the responsibility for data retention and structure, and Sun ONE only provides a new method for viewing or analyzing the data. SOAP is reliable in providing a data-viewing interface, similar to the way in which other Internet protocols address their specific tasks, such as limited data retrieval with HTTP. Sun ONE will not improve the accuracy of data.

In some cases a project may be better handled through another aspect of service-oriented architecture. SOAP, for example, does not offer easy scalability for high-traffic applications unless properly managed. Enterprises that need such scalability may prefer to completely replace an established legacy data base structure to avoid problems associated with Sun’s relative immaturity as a Services-On-Demand vendor.

Sun ONE-enabled data interfaces will not make the data any more accurate or increase its usefulness. Ways exist around the accuracy problem using search and associated technologies, but they can add dramatically to costs.

Although RTE projects can save money by making repositories and data logic reusable, organizations should not force Services-On-Demand onto all RTE-related projects. The general usefulness of Sun ONE, however, makes it particularly suited for many aspects of application integration. Failing to use Web services where it is the best solution would be a costly mistake.
The biggest concern among Web services users is security. A recent survey, "Enterprise Development Management Issues 2002," by California-based market researcher Evans Data, found that security and authentication were the top hurdles for 48 percent of the 400 respondents.

Security actually poses separate problems: one technical, one business-based. On the technical side, there are no true security standards for XML. Even if there were, what would the contract language (the “business” issue) need to be to convince CIOs that they could safely use a company that their Web services found on the Internet for important transactions? Who do users trust--and how do users validate that trust--in the ideal world of a Sun ONE-based Web Service talking to a Microsoft .Net-based Web Service?

Until these problems are solved--which will not be for another couple of years--firms hoping to use Web services must establish their own security and reliability solutions. In addition, while it is simple to convert an existing application into Services-On-Demand, adopters must be careful as anything done now will have a big impact on their overall systems architecture in the future.

While it is just a matter of time until standard security protocols emerge, it is enough to give anyone trying to use Web services a headache. Currently, every XML security protocol on the market is a proprietary vendor offering and therefore not truly open. Hugo Haas, Web services activity lead with the Cambridge, MA-based World Wide Web Consortium (W3C), the standards setting group, says that at this point the W3C has not even finished determining everything an XML security standard would require, let alone deciding on a standard. Until the security issues are cleared up, the one-time transactions that would come from a Web services "yellow pages" (UDDI) are a fantasy at best.

Joseph Williams, global chief architect for Sun One professional services in Denver, summed up Web services' other weaknesses as "all the '-ilities," listing reliability and scalability as the chief culprits, but leaving the door open for others.

Reliability and its attendant weaknesses have a common cause: the Internet. Specifically, the HTTP communication standard, which just is not a good fit for Web services. The big issues with HTTP are that it is connectionless and eventless, and it can not handle distributed transaction coordination the way common object request broker architecture (CORBA) can. In the middleware platforms of CORBA and COM, tightly coupled, internal software development worlds, data delivery is guaranteed. The Web services world is loosely coupled, which means that delivery is not always guaranteed. Ted Schadler of Forrester Research compares HTTP to a telegraph line and says that the machine-to-machine transactions in a Web services world require an open-line, telephone-like connection. Unfortunately, no one has figured out how to do that yet.

HTTP works pretty well most of the time, with the exception of when users are considering sending sensitive, unreliable business data. Although HTTP is reliable 99 percent of the time, there is a possibility that users could lose packets. For financial and other critical data, these odds are too risky.

Despite successful workarounds and inevitable solutions to Web services' shortcomings, it is relatively easy to convert an existing application into a Web service. CIOs should err on the side of caution here. While on the surface Web services seems to eliminate the need for middleware to connect applications and data bases, that is not always the case. In fact, the opposite is true: Web services is simply another layer that uses middleware to connect to the data. Building a lot of Web services could prove fatal to a company that does not have a middleware-intensive, three-tier architecture in place.

Furthermore, Web services adds an extra layer of complexity that not all IT departments are in a position to handle. When an application fails, the application administrator blames the hardware administrator, the hardware guy blames the application administrator, and everyone blames the data base expert. In the brave new world of Web services, blame can be placed on the Web services layer. Building a plethora of Web services applications before installing an inventory system to keep track of them makes them impossible to reuse--which is the reason for building Web services in the first place.
Microsoft and Sun are arguably the two most important companies in the Web services arena. In fact, Microsoft's .Net has become so synonymous with the generic term Web services that some people think they are one and the same. That notion is emphatically false.

The cure for anyone that believes Web services equals .Net is Sun Microsystems, at the head of the Java-inclined crowd with its Sun ONE.

A not-so-recent reshuffle of Sun's iPlanet, StarOffice, Chili!soft, Forte, and other offerings has drawn them under the Sun ONE umbrella. This repackaging is as much an exercise in promoting the Sun ONE brand as it is in simplifying and developing the product range.

Branding and buzzwords aside, what are the significant differences between Microsoft .Net and Sun ONE? Just as users, analysts, and the media have different interpretations of Web services, so do vendors--in fact even more so in this case.

Take these simple goals: Microsoft wants to ensure Web services are prefixed by XML. Sun wants to see Web services hosted on Java. Both are moving out of their comfort zones. Microsoft must look more than it has in the past at back-end systems, moving toward server software and further away from the PC software that has historically generated the bulk of its income. Sun, which makes a good living selling hardware, must stop thinking about server boxes and workstations and concentrate on software packages.

Due to its Java heritage, Sun focuses on platforms and systems above all else. Microsoft worries more about interoperability. Sun talks about integration, but makes its living selling hardware. Until recently, it did not focus on software at all.

While the idea of interactive applications--like those that make travel arrangements, arrange for the appropriate foreign currency, and reschedule diary appointments as soon as users log on and purchase a flight--may be years away, this is the all-encompassing, consumer-embracing vision that Microsoft is targeting. Although there are still reliability issues, and the delivery timeframe is uncertain, the vision is there.

Sun, on the other hand, is leading with the technology mission rather than the user experience. In other words, it is working to transform B2B interaction in the back room, under the covers, far from things most users will ever see.

Though the vision may look less complete and the back-end orientation less glamorous than Microsoft's, the technology components are at least as credible. Sun's strength lies in back-end applications and integration, not in the client interface. Obviously, most business entities will require both.

Sun has been perceived as having an isolationist stance, and has been criticized for that. Many industry insiders feel that Sun has tried to go it alone with Sun ONE, leaving it on the sidelines while Microsoft and IBM ploughed ahead with a consensus on standards.

Simon Phipps, chief technology evangelist at Sun, defended the company's position like this: "Microsoft and IBM chose not to invite us, chose to make up lies about Sun not being committed to Web services. If we were Web services laggards, we wouldn't have been committed to SOAP with IBM in 2000." All claims, counterclaims, and name-calling aside, keep in mind that Web services have barely left the nest.

At the moment, it is best to think of Sun as having powerful tools for back-end, server-side Web services creation and deployment, while Microsoft excels at user-friendly, visible, client-side desktop software. It seems certain that as Web services grow toward their full potential, there will be room for both.
Recommendations

If no previous systems exist to deal with, businesses can use Sun ONE as a primary development element. A "green-field" project presents an enterprise with the opportunity to develop a services orientation without requiring tools or connectors for legacy applications and architecture. Links between enterprises are easily made. Green-field projects are attractive because no reliance exists on wrapping or incorporating established technologies. The newness of the environment invites novelty in the solution. New products and frameworks can give all users access, regardless of their PC, operating system or location. For example, i-Deal is using Web services to provide a neutral platform for connecting broker/dealers in the financial industry.

For uBid, an online auction powerhouse, the benefits have been dramatic. It now takes only two weeks to add aggregators, or auction "middlemen," to its system. Buyers, of course, have instant access. Enterprises should not earmark strategic or highly visible projects for Web services unless their corporate culture is suited to taking risks with IT. Selecting strategic projects for Web services will almost certainly result in managerial disappointment. For most enterprises, tiny steps are the best way to start. Web services, however, should not be used to fix problems that will not affect the business. This will associate them forever with unremarkable projects.

On a small-scale project, learning to deploy Web services will not delay the project too much. Astute enterprises will use Web services to address problems where a solution will get attention, but is not eagerly awaited or business-critical. For example:

A Web services implementation at the Colorado Department of Agriculture led to substantial reductions in some data processing times. Field workers now have immediate access to the latest data instead of having to wait weeks for a report.

A sales data analysis system at Rotech Healthcare makes it possible for Rotech to generate readable charts and graphics from previously incompatible data sources, giving it immediate insights into the effects of potential changes.

Use Web services for the following:

- New projects with no legacy issues or applications to incorporate.
- Non-invasive integration of legacy systems where projects are tactical but popular and the data is clean, intelligible, and can be used in multiple projects.

For the next two years, most enterprises should limit their use of Web services to projects that are not time-critical, but will be received positively. Wider implementation will likely incur significant risks that the risk-averse majority should shun. Wider implementation will likely result in ill-considered failures and internally prominent examples of waste.

Assuming that the current level of industry cooperation continues, the current performance and security limitations should not hold back Web services for long. Forrester's Schadler says that the Internet makes Web services so cheap that whatever problems there are now will eventually be solved. "Saying people won't use Web services over the Internet because of HTTP is like saying no one will use e-mail because the addresses are too hard to remember," he says.

About the Author

Dean Calabrese is a Senior Project Consultant. He has previously served as the President of Spider Partners, VP of Application Development for CBS and International Creative Management, and CIO of CBS/Fox Video and Charmer Industries.
Web Links

IBM: http://www.ibm.com/
Microsoft: http://www.microsoft.com/
Sun Microsystems: http://www.sun.com/
Universal Description, Discovery, and Integration (UDDI): http://www.uddi.org/
World Wide Web Consortium (W3C): http://www.w3.org/