SearchWebServices.com E-Guide

ESB E-Guide

The enterprise service bus is a key element in a service-oriented architecture; and often times, it can be used as an incremental starting point to implementing SOA. This E-Guide provides an overview of the different ESB vendors and options; details how the ESB can provide a "simplifying" façade to the complexity that otherwise would be implemented in every application or service in a SOA; and covers common mediation and orchestration requirements and how ESB and BPEL implementations can help satisfy those requirements.

Sponsored By: Progress Sonic

SearchWebServices.com E-Guide

ESB E-Guide

Table of Contents:

ESB market report

By Colleen Frye, News Writer
26 Oct 2006 | SearchWebServices.com

This bandwagon's a bus and all walk of vendors positioning in the SOA market have jumped on it over the last few years.

The bus, of course, is the enterprise service bus (ESB) and as it has gained momentum as a key element in a service-oriented architecture. The market has expanded from the domain of startup specialists to integration and infrastructure players and platform vendors as well as the open source community. According to a recent Gartner Dataquest report, the ESB market expanded more significantly in 2005 than any other application integration and middleware segment, with 160.7% revenue growth.

In addition, Cambridge, Mass.-based Forrester Research expects to see 67% of firms with 40,000-plus employees implement SOA this year and found that 83% of firms are using SOA for internal integration. These trends will help drive adoption of ESBs, according to Forrester in its recent Forrester Wave: Enterprise Service Bus, Q2 2006.

According to Forrester and other industry experts, many organizations implement an ESB as an entry point to SOA, but they all caution that an ESB is only one piece of an enterprisewide SOA and there are alternatives to consider.

"The real question is, how can an ESB help with a SOA implementation plan, which is the challenge a lot of enterprises face," said Jason Bloomberg, a senior analyst at ZapThink LLC in Waltham, Mass. "Companies need to think through a plan, identify the business problem, figure out how to build services, then figure out the infrastructure. They may or may not need an ESB."

Starting point

Today, though, many enterprises do start with an ESB, which Forrester principal analyst Ken Vollmer describes as a straightforward way to get started with SOA and typically a less costly route to application integration than integration-centric business process management (BPM) suites. Vollmer identifies three types of ESB vendors: pureplays, those with roots in enterprise application integration (EAI) and platform players. In addition, there is a handful of open source ESBs to choose from, both standalone and as part of SOA platforms. Also, vendors like Microsoft and SOA Software Inc., for instance, don't offer ESBs per se, but do have ESB-like functionality with their offerings.

According to Forrester, an ESB typically includes communication infrastructure; request routing and version resolution (mediation); transformation and mapping; service orchestration, aggregating and process management; transformation management; security; quality of service; service registry and metadata management; extensibility for message enrichment; monitoring and management; and support for the service life cycle.

The early ESB market, which was based on Web services standards, stole some thunder from the EAI market, which was saddled with proprietary technology. "If you look back five years, the ESB vendors were putting the hurt on application integration vendors," Vollmer said. "But the EAI vendors moved up the stack by adding more advanced BPM features and they've adopted the same standards as ESBs. You could say now that integration-centric BPM could commoditize the ESB market to some extent."

But by far the biggest change in the ESB market, he said, is "you find more and more larger vendors adding either a standalone ESB, like IBM and BEA, or including those features in basic integration suites, like Tibco and WebMethods and Oracle. Over time, I believe standalone ESBs will gradually be consumed by the larger vendors and become more basic infrastructure technology," Vollmer said.

A maturing market

As the larger vendors have added ESB capabilities to their offerings, the pureplays have broadened their capabilities as well. "Look at Sonic Software," said Bloomberg. "They're still leading with their ESB product, but Progress Software [Sonic's corporate parent] has reorganized. It's significant that Progress has realized SOA is more than ESB. They have the Neon legacy integration, they have XML tooling. SOA requires a lot of different pieces. Sonic's leading standalone ESB product is really not standalone anymore. Companies want a more comprehensive solution."

According to Kelly Emo, product marketing manager for San Jose, Calif.-based BEA Systems Inc.'s AquaLogic Service Bus, "Expectations are changing as customers become more mature in their approach and implementation of SOA. One of the biggest changes is the expectation that ESB plays a critical role in the overall service lifecycle. They're interested in a service integration backbone, but having it be a participant, from designing, discovering, integrating and managing services. A lot of requirements come to us about integrating what the bus knows about services in the registry/repository."

Today a small percentage of customers specifically ask for an ESB, said Ashish Mohindroo, senior product director for Oracle Fusion Middleware, at Redwood Shores, Calif.-based Oracle Corp. However, said Mohindroo, "the majority will come in saying they're trying to move toward SOA and they need something lightweight to do the transformation and routing. But the more we explore ESBs, they realize it's just a small component in moving toward SOA. You also need orchestration, security, monitoring, and governance."

Intermediary alternatives

ESBs are just one type of SOA intermediary; other types are Web services management products, XML gateways, pureplay mediators like SOA Software and Apache Synapse, and platforms, according to Anne Thomas Manes, vice president and research director at Midvale, Utah-based Burton Group Inc.. She believes ESBs are best suited for complex aggregation and transformation of data, legacy access and orchestration. "I don't typically recommend using an ESB for mediation, they're more in the platform category," she said. "I typically recommend an XML gateway or Web services management product. Both have stronger security mediation."

According to ZapThink's Bloomberg, "A key part of the SOA infrastructure has to do with the intermediary capability for loose coupling of services. If your existing middleware can't do that, bring in some intermediary. If you have the need for additional integration infrastructure, get an ESB. If you already have that infrastructure you can use an intermediary, like SOA Software's Network Director. The question is, do I need messaging infrastructure in addition to intermediary capabilities?"

Oracle's Mohindroo agrees that there are alternatives. "Some customers don't need a pure ESB environment. There are emerging standards [like WS-Reliable Messaging]. So if you don't have too many services or systems to inter-operate, you can get away without an ESB and make do with an orchestration engine, say Oracle BPEL Process Manager. You can connect services and define the flow. We have several customers today that leverage that, who don't have those requirements that a typical ESB would provide."

Manes anticipates that as XML gateway providers add more ESB functionality into their boxes and once those boxes support WS-Reliable Messaging and routing of events and more complex integration, "I don't see a whole lot of value in the ESB as an intermediary." Where she expects to see the ESB evolve, and continue to have a role, is as a platform.

Oracle's Mohindroo agrees. "With ESBs, think next-generation messaging bus. After you connect [the systems] what do you do? The ESB becomes part of the platform to access these different services. Most companies are looking toward the ability to build flexible infrastructure. Think of the ESB as an entry point."

What Oracle, IBM, BEA and other platform players are hoping, is the ESB entry point will lead to adoption of an SOA platform. "What we're seeing is a big adoption of the SOA platform, which naturally leans to a bigger platform provider," said Mohindroo. "We're seeing customers ask for an end-to-end SOA platform, for technological reasons, but second, for what kind of support we can provide. All those aspects become really critical when you talk about SOA infrastructure."

New entries

Open source vendor JBoss, now part of Red Hat Inc., is the latest platform provider to throw its hat into the ring with its JBoss Enterprise Service Bus, now in beta. And IBM, which also entered the ESB market later than some of its competitors, has two ESBs: WebSphere ESB, and WebSphere Message Broker, which the company has repositioned as an advanced ESB.

"We have had our Message Broker in the market for a long time. September of last year ESBs became hot, that's when we introduced WebSphere ESB and the Message Broker portfolio," said Sandy Carter, IBM vice president for SOA and WebSphere. Carter said WebSphere ESB is suited for organizations looking for Web services connectivity and Message Broker is for those organizations that also need integration with non-SOA environments.

"ESB is at the heart of our reference architecture for SOA," Carter said. "Having a strong ESB, we believe, adds tremendous value to what you have in the marketplace. The power of SOA is not just to do new things, but you can integrate what you have in your environment, and reuse it, and have a high quality of service. The way to get this is through this ESB layer. We believe it's a real powerful part of the SOA story."

Emphasis on “part of the story” according to most observers.

BEA’s Emo said that ESB capability should be a factor in choosing an SOA solution, “but on a broader basis. The best recipe for looking at SOA is to look at your lifecycle needs. It’s important to have the core features of an ESB, but not in a vacuum.”

“What we’re seeing from ESB vendors as well as consultants building SOA solutions, is that the bloom is off the ESB rose,” said Bloomberg. “Clearly the role of ESB is no longer thought of as a key piece of SOA infrastructure, but rather one piece of many moving parts to get SOA to work.”

We believe that the right infrastructure for SOA delivers success at every step.
Now that's Progress.®℠

We see a world in which the architecture of business applications, and the infrastructure that supports them, no longer create rigid business systems and change-resistant processes, but instead are able to support the timely execution of business decisions.

In this world, agile businesses respond more quickly to market changes and customer demands, adjust their business practices in real time, and deliver new products and services faster than their competition.

Progress provides infrastructure products and best practices that enable companies to lay the flexible, reliable, scalable and manageable foundation required to deliver on the promise of SOA *today.*

Progress Sonic ESB products, recognized as #1 in the market, simplify the integration and flexible re-use of business applications within a service-oriented architecture.

Progress Actional SOA Management products provide operational and business visibility, policy-based security and control of services and end-to-end business processes in heterogeneous environments.

Progress DataXtend products offer a unique approach to data management, including employing a common semantic data model to facilitate the integration of heterogeneous data sources.

So if you share our vision, welcome to our world. Achieve the promise of SOA today and rapid ROI.

Call us at +1 866 438 7664 or learn more at:
www.progress.com/believe

Change is good. Progress® is better.℠

PROGRESS
SOFTWARE

Using an ESB to simplify the complexity of SOA

By Maja Tibbling, Lead Enterprise Architect, Con-Way, Inc.
20 Mar 2007 | SearchWebServices.com

Why add yet another moving part into the service-oriented landscape? Isn't management of service-oriented applications already too complicated? The reasons for introducing an enterprise service bus are the same as those applied when choosing to implement an enterprise application integration strategy some years ago.

In the early stages of an SOA implementation, when the inventory consists of nothing more than a handful or two of project-based services, an ESB seems to be overkill. With any luck, however, the service deployments will accelerate as adoption within the enterprise progresses. A strategy is required to provide on-demand scalability, reliability and adequate performance characteristics. From an architectural perspective it is also a good idea to have an organizing principle to avoid service chaos.

As an enterprise SOA matures, business functionality will be mined and exposed from a variety of sources. These service providers may be legacy applications, third party packages or capabilities offered as hosted solutions. While the ideal would be to have all services using the same technology, the real world has proven that this is unrealistic. Web services standards will most likely be just one of many approaches used.

Advanced SOA functionality, such as service orchestration, automated business processes, event-driven architecture and complex event processing, is dependent on sound enterprise application integration architecture.

With these considerations in mind, the justification for the implementation of a full-featured integration broker with ESB capabilities becomes clear.

Reliability and Availability

Visibility to all the moving parts in distributed software has been a major challenge since the advent of n-tier applications. With the complete separation of concerns in service-oriented development of applications, a prospective consumer of a service must be able to find it and know what to expect for availability, quality of service and performance characteristics. The ESB works with the service registry as well as helping with the compliance with SLAs -service level agreements. In order to monitor the health of a service, dependencies on other services, as well as the state of the execution platform must be known. When problems are detected, alerts must be raised and notifications sent. Many integration broker vendors provide either built-in monitoring capabilities or, more commonly, hooks to monitoring tools for more complete visibility. Common services for audit trail, logging and exception handling can be hosted by the tool.

Scalability

The bus-based infrastructure provides an abstraction from the actual deployment topology of the services. Capacity can be added to the Web server, application server, third party package, legacy app and database instances that are doing the actual work promised by the service descriptions without impact to the service consumers. In fact, work can be delegated to federated buses, so the ESB itself can be deployed in a manner to make the best use of hardware and network resources.

Security

An implemented service should not bear the burden of managing which consumers and under what conditions they have access. Those rules are likely to change over time. This cross-cutting functionality is externalized from the service in the ESB.

Extensibility and Agility

As business is constantly adapting to changing times and new opportunities, the business processes will be evolving as well. Planning for flexibility is critical for a successful SOA implementation. Transformations should be done as close to the integration endpoints as possible to and from a canonical (common) business information model. This approach will facilitate agile automation of processes, orchestration of composite services and externalizing of process-specific business rules from the business functionality of the service.

In conclusion, a full-featured integration broker will provide core ESB capability as well as all other functionality required in a heterogeneous service topology. This class of technology provides a “simplifying” façade to a great deal of complexity that would otherwise have to be implemented in every application or service.

SOA infrastructure: Mediation and orchestration

By Satadru Roy, Technology Architect
07 Nov 2006 | SearchWebServices.com

There seems to be no end to the hype surrounding SOA. Vendors and analysts are only too happy to fuel the media buzz by touting their own products and research recommendations. Some claim their products go a long way towards enabling SOA while others maintain that SOA is an architectural pattern, albeit one also better implemented with their offerings.

No matter what claims are made in the marketplace, it is important to realize that any one SOA infrastructure product represents a means to an end, not necessarily an end in itself. Platform software can provide the necessary backbone for an enterprise-wide services fabric, but the choice of the solution should be solely dictated by the requirements of the organization.

One such software product is the now ubiquitous enterprise service bus (ESB). Few recent innovations have generated as much excitement and confusion. All ESB vendors claim that their enterprise service bus implementations provide a key piece of the service-oriented technology architecture. However, most still struggle to agree on what actually constitutes an ESB, let alone what capabilities the common ESB should have.

Many other vendors are jumping on the ESB bandwagon. Traditional EAI products are being rebranded as ESBs and pure-play messaging vendors are doing the same by implementing a SOAP and WS-* stack on top of their existing solutions. Some vendors are even offering not one, but two sets of ESBs!

Imagine, amidst all of this, that you are the IT architect charged with the responsibility of 'SOA-enabling' your applications. What should you do? It's simple really, just go back to your business requirements as they can be fulfilled by service mediation and orchestration.

Service mediation

The concept of mediation is nothing new. In traditional object-oriented literature, a mediator is a well-known pattern that promotes loose coupling by keeping objects from referring to each other explicitly and lets you vary their implementation independently.

Replace the word objects in the above sentence with services and you will have a good starting definition of service mediation. However, service mediation in an SOA also goes much further than merely keeping services from referring to each other (although that in itself is a good start). In the world of services, where the emphasis is more on technology-neutral, XML-based message interactions, a number of things become possible once you put a mediator in between a service producer and a service consumer. For example:

Transport protocol conversion

Often, the service provider offers a service based on a certain transport protocol (such as HTTP) whereas the service consumer is only capable of communicating over a different protocol (let's say MQ). Alternatively, perhaps a synchronous-asynchronous bridge needs to be built connecting a service provider and consumers over protocols such as HTTP and Java Messaging Service (JMS). Technically speaking, JMS is not a transport protocol, but rather a vendor-neutral set of messaging APIs. As illustrated in Figure 1, asynchronous Web service implementations often expose "SOAP services over JMS," but under the covers the transport protocol used is vendor-specific, such as MQ or Tibco RV.

This is quite common in environments where some legacy assets may have been service-enabled, but other applications cannot consume those services because of a transport protocol mismatch. Rather than building yet another protocol adapter or implementing a synchronous-asynchronous bridge on top of the service provider implementation, it is better to let a mediator handle these differences and do the necessary translation. The service developers can then focus on building the service logic, letting the infrastructure software handle mediation responsibilities.

Data format conversion

Even within the same organization the business entity definitions may vary from one organizational unit to another. The finance department may have a specific customer structure that could be different from the definition of a customer from a billing perspective. In this situation a customer-related service in billing cannot consume a service from finance without having to account for the data model differences.

You can try and force everyone to adopt a common definition (as with the proposed Canonical Data Model), but it may be impractical to do so in a large organization. What is preferable is to let a mediation component handle the transformation between one format and another. It may even be possible to let service interfaces deal only with a canonical data model, but service consumers will then need to be built with mediation components to transform their data format to the one expected by the providers.

Service policy enforcement

Having a mediator handle the interaction between the service provider and the consumer allows it to easily intercept XML traffic between the two and take necessary steps to ensure policy compliance—in the form of appropriate auditing, logging and security monitoring, for example.

Once again, delegating these cross-cutting concerns to a common mediation component frees up the service developer from having to fulfill their requirements in the service level. Of course, if this is the only mediation requirement you are faced with it may be more cost-effective for you to implement a cross-cutting solution of your own based on technologies such as aspect-oriented programming (AOP).

Service processor pipeline

The pipes and filters architectural pattern describes how general-purpose message pre-processors and post processors can be built to enhance a message processing cycle. A mediation platform can compose such filters in a declarative fashion for a service invocation allowing us to vary the invocation sequence of these pre and post processing components by changing the configuration of the filters.

Examples of these pre-processors include service routing based on message content, context or business rules, message enrichment, message encryption/decryption, message de-duplication, and so on (see Figure 1). The key value-add of mediation here is not so much in being able to provide such processors which developers have to build anyway but to facilitate an arbitrary composition of these processors in a declarative fashion.

Service invocation and dispatch

Remember the part about loosely coupling the service provider and the consumer? If the service consumer interacts directly with the service provider it may be difficult to change the interface (not the implementation, which should be independent of the interface anyway) of the service provider without impacting the service consumer. In other words, the service consumer forever remains tightly coupled with the service provider.

However, if the service consumer interacts only with the go-between mediator and that mediator service does not change its interface the mediator can transparently dispatch the service invocation to different versions of the service, possibly even with different service interfaces. Obviously, the mediation (the necessary translation) in this case still needs to be built, but the important point is the independence achieved between service consumer and provider.

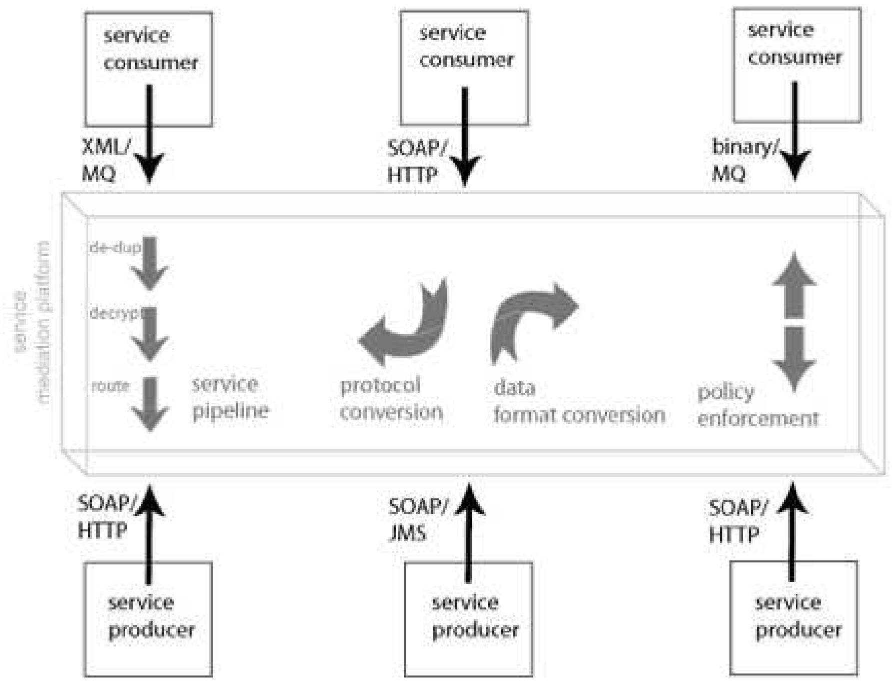

Figure 1: This diagram illustrates how the previously described requirements can be handled by a common mediation layer.

The enterprise service bus can be viewed as a mediation platform supporting some or all of the capabilities we just described. In fact, you will likely find even more features in some of the higher end ESB products.

Service orchestration

Unlike service mediation, which essentially establishes a broker component as part of contemporary middleware, orchestration provides capabilities associated with the composition and coordinated execution of services. Orchestration technology is also middleware-based and can establish a highly centralized part of the architecture that governs the design of business process definitions, as well as the execution of business process logic.

Service layers

Before we describe orchestration in detail it is important to understand the concept of service layers. Note that this discussion serves only as an introduction to service layers and domain abstraction.

Service layering is a separation of different categories of services (called service models) based on the problem domain they operate in. Business services, for example, focus solely on business logic whereas application or utility services are more technology-oriented. Examples include services that facilitate solving system-level problems such as security and auditing. Figure 2 (reprinted with permission from "Service-Oriented Architecture: Concepts, Technology, and Design" by Thomas Erl) illustrates the relationship between common service layers.

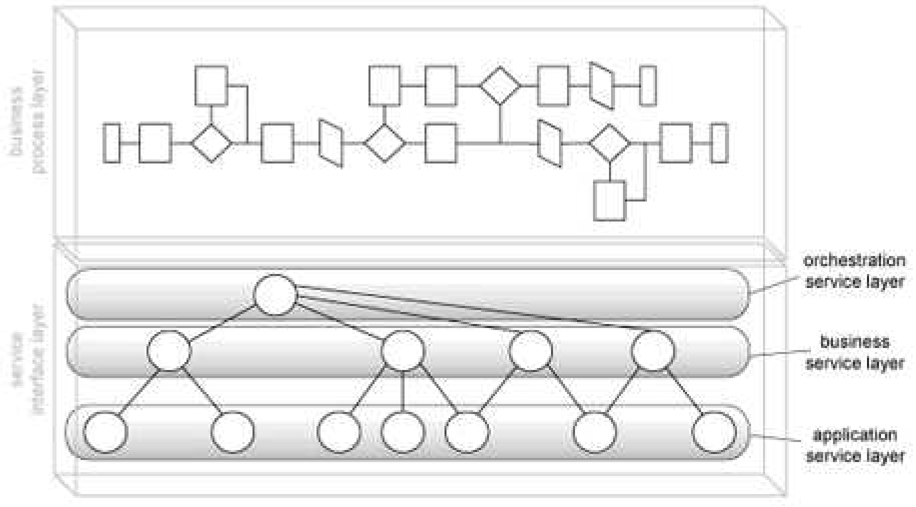

Figure 2: The three common service layers that are implemented through the use of service models.

Business service implementations are likely to make use of utility services, and possibly other business services as well—this is known as service composition. One of the fundamental principles of service design is that a service should be built such that it can participate in a composition, if necessary.

Service orchestration is an extension of the service composition model where a parent service layer performs an orchestration of many business and utility services by controlling workflow logic and invocation sequences.

For example, a mortgage solution may need to make use of credit score services offered by external credit agencies. The calls to the credit agencies can happen in parallel, but the results of the calls may need to be aggregated and analyzed to arrive at a decision. Such complex workflow logic often involves timeouts and exceptions and is typically modeled as business process definitions.

Service orchestration and BPEL

The key factor in determining whether you need a true service orchestration platform is figuring out if your service and business process interactions require a complex pattern of invocations as described above. If they do, then you might want to consider implementing them in a service orchestration platform such as a Business Process Execution Language (BPEL) engine.

BPEL has emerged as the most promising standards for Web service orchestration and even though it is often mentioned in the context of modeling executable business processes, the service orchestration aspects of a BPEL engine cannot be overemphasized. It is important to keep in mind that BPEL has limited capabilities in terms of modeling real-world complex workflows and as such it may be better to focus on BPEL more as a service orchestration vehicle than a full-blown workflow modeling tool.

BPEL and ESBs

Some ESB products come packaged with a BPEL implementation while others do not. The ones that don't may provide a graphical wizard based on service mediation flow constructs to model simple short running service flows. If you are tempted to model all your short running, synchronous service flows and interactions via such tools think long and hard about the true mediation requirements you have. If you can't come up with many, then a code-based implementation, rather than investing in expensive middleware, may be a more prudent choice.

Additional orchestration capabilities

Provided below is a brief list of other features and capabilities provided by commercial orchestration products:

- Management of synchronous, stateless (short-running) flows with complex interaction patterns, such as parallel invocations and time-bound executions with possible cancellations.
- Management of long running, stateful, asynchronous flows where processes may need to wait for events to fire. For asynchronous invocations, callbacks with correlated request-response patterns may also need to be supported.
- Parallel service invocations with support for AND-join and OR-join conditions.

Orchestration technology has been around since the days of EAI. Therefore, it has matured and become relatively commonplace. However, when assessing an orchestration product for use within a service-oriented technology architecture, it is well worth investigating the extent of support it provides for contemporary Web services technologies and the common principles of service-orientation.

Future trends

Even as commercial ESBs acquire more sophistication, some open source mediation frameworks are gaining traction and are solutions you might want to consider, especially if you are building using the Java Enterprise Edition (JEE) platform. The most popular of these frameworks offer mediation functionality such as protocol conversion, data transformation and service routing. However, keep in mind that these may not provide the rich user interface offered by the commercial products.

The other interesting trend is the very recent emergence of what can be characterized as hardware ESB appliances. These are primarily positioned as "edge-of-the-enterprise" XML security devices, but they also offer strong mediation capabilities. Furthermore, because the implementation consists of actual dedicated hardware, these devices can offer superior processing performance. However, it remains to be seen if they gain widespread adoption or even supplant the software ESB solutions. We will look at these appliance solutions in more detail in a future article when we discuss SOA security infrastructure software.

This article was originally published in The SOA Magazine, *a publication officially associated with* "The Prentice Hall Service-Oriented Computing Series from Thomas Erl". *Copyright © 2006* SOA Systems Inc.

Resources from Progress Software

The Right Infrastructure for SOA

The Sonic ESB: An Architecture and Lifecycle Definition

Faster SOA Integration with Progress Sonic ESB 7.5

About Progress Software Software

Progress Software Corporation (PSC) (Nasdaq: PRGS) supplies industry-leading technologies for all aspects of the development, deployment, integration and management of business applications. PSC, headquartered in Bedford, Mass., operates through the Progress Company, Sonic Software Corporation, PeerDirect Corporation, and DataDirect Technologies. PSC can be reached at www.progress.com or +1-781-280-4000.