SONICMQ — THE ROLE OF JAVA MESSAGING AND XML IN ENTERPRISE APPLICATION INTEGRATION

Progress Software Corporation

October 1999

SonicMQ — The Role of Java Messaging
and XML in Enterprise Application Integration — Progress Software Corporation
is published by Hurwitz Group, Inc.
111 Speen Street, Framingham, MA 01701
Telephone (508) 872-3344; Fax (508) 872-3355
Email address: info@hurwitz.com
Web site: http://www.hurwitz.com

October 1999

CONTENTS

EXECUTIVE OVERVIEW

Enterprise applications must meet demanding requirements: performance, reliability, flexibility, and ease of use. Information must be integrated from disparate, loosely coupled systems — both within a business and between businesses (B2B) to support electronic business. Doing business over the Internet demands high performance, low latency, and reliable data exchange across large networked systems within the enterprise and B2B.

What's needed is a message-oriented middleware (MOM) solution that can support the following:

- A high-level applications interface (API)
- Quality of service (QoS) guarantees
- (Near) real-time application integration
- Event-driven processing
- Security
- Management of message traffic

The Java Message Service (JMS) can provide the API and basic messaging services (e.g., message queuing, publish/subscribe). But to deliver a complete infrastructure that will meet MOM requirements for enterprise application integration (EAI) and e-Business, vendors will need to extend the JMS specification and add capabilities for security, QoS, system administration, and workload distribution. To promote data exchange between disparate systems, support for additional data formats, especially XML, is also desirable.

This white paper explores these requirements and shows how one vendor, Progress Software, is providing a solution through its JMS-compliant SonicMQ product.

INCREASING DEMANDS ON ENTERPRISE-LEVEL COMPUTING

Today's enterprise-level systems have evolved from older stovepiped systems (standalone applications that do not integrate with or share data or resources with other applications). By their nature, standalone business systems do not interoperate, and they therefore create the additional problem of needing to duplicate business data. Standalone systems make it difficult to view the same data from different perspectives, for example, to recognize patterns in customer and vendor behavior, or to bring information together about customers and vendors (buyers and sellers). Even client/server (C/S) systems, which can support integration within a single department or organization, make it difficult to exchange information across department boundaries.

The Internet has forced a change in this picture. The Internet has promoted the globalization of business and made it critical to interconnect corporate offices and organizations. Both data and business processes need to be shared. Infrastructure services need to be consolidated and centrally managed. Business logic needs to be reused, changed quickly, and redeployed. The diversity of enterprise-level systems and their distribution across geographical boundaries demand open services, APIs, and protocols to exchange data efficiently.

MESSAGING REQUIREMENTS FOR SUPPORTING E-BUSINESS

Success in this new competitive environment is directly linked to a company's ability to make it easier and more profitable for customers, suppliers, and partners to do business with it. This ability will depend on a company's computer-based support for information handling. At the most fundamental level, the primary technical challenge of e-Commerce is routing messages reliably to and from the appropriate systems. Messaging middleware products, with software that provides an interface between applications, allow systems to send data back and forth to each other asynchronously. Messaging-oriented middleware (MOM) can coordinate the message traffic between components of distributed systems by providing lower-level, but still mission-critical, services such as message queuing, QoS guarantees, and transaction support.

To an e-Business vendor, effective MOM ultimately assures that information flow is fast and reliable between systems that take incoming orders and those systems that handle order fulfillment and accounting.

DISTRIBUTED COMPUTING AND IMPLICATIONS FOR DATA EXCHANGE

Commonly used methods for exchanging information between two applications include peer-to-peer and Remote Procedure Calls (RPCs), CORBA, COM, and RMI. These methods have had their place in tightly coupled computing models; CORBA and COM both bind distributed objects tightly together. Tight coupling itself introduces further problems; for example, it is harder to make

changes in one component without introducing side effects in others, and version change and version tracking become a headache. These methods also require information to be reduced to a block of data before the data is passed. And data blocks are by themselves not self-describing. Developers must embed descriptions of data structure and content in the code — this means longer development times, extra lines of code, and large portions of code devoted to validating information contained in the data.

Distributed architectures also introduce problems for components that must communicate because components can fail in several ways by:

- Going offline without warning
- Terminating unexpectedly
- Not responding to a request quickly enough

Messaging services, especially MOM and JMS, are designed to address these problems.

COMMUNICATION AND MESSAGING SERVICES

Middleware

Middleware is connectivity software. It has often been called the "glue" that provides for communication across heterogeneous platforms. Middleware consists of enabling services that let processes running on diverse machines interact across a network. Middleware has enabled interoperability for C/S architectures. DCE, CORBA, and COM are all widely-used middleware frameworks.

Message-Oriented Middleware (MOM)

MOM is a specialized class of middleware that extends process-to-process communication in a distributed environment. It provides message passing or message queuing services and uses queuing to support asynchronous communications (i.e., client messages are sent to a queue until retrieved). The advantage of this scheme is that the receiver can be offline when the message is sent. The order for retrieving messages from a queue is open. Hence, MOM can be used with load-balancing and prioritization schemes for message delivery. With MOM, fault tolerance can be provided because persistent queues let messages be recovered in the event of system failure.

- MOM is preferred to other communications services, especially RPCs, for corporate intranet and web-based messaging (see Figure 1). MOM is primarily used to support deferred communication, whereas peer-to-peer and RPCs are used to support synchronous

communication. With RPC, the receiver must be online. But with MOM, the sender can send messages to receivers (e.g., servers) that are down at the time the message is sent.

- MOM decouples resources. It promotes building loosely coupled systems and thus helps developers build highly modular systems. Component reuse and reliability are increased in so far as a failure in one component is less likely to affect another.
- MOM views messages as events rather than as method calls (RPCs). Clients can send and receive events (messages) via APIs that MOM provides. Applications or application components communicate with each other using MOM.

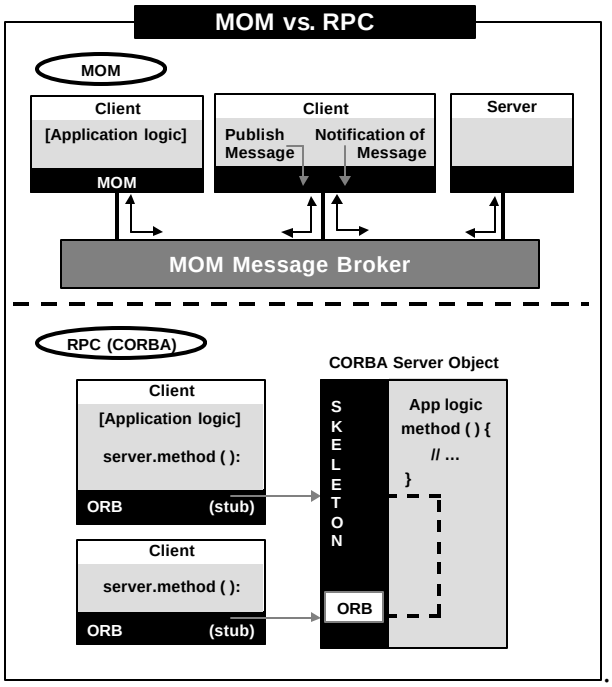

Figure 1. MOM vs. RPC

The Java Messaging Service (JMS)

The JMS is an API for accessing enterprise messaging systems. The JMS specification defines an interface but does not itself define an implementation. The specification is vendor neutral — it sets requirements but does not dictate how they are to be implemented. This means that it is up to the vendor to add facilities, services, or enhancements not included or defined explicitly in the specification.

- JMS does *not* include load balancing/fault tolerance, error/advisory notification, administration, or security. These facilities are, of course, the "must-haves" for corporate intranet and B2B communications with e-Commerce. Vendors can differentiate themselves by adding just these functions to their JMS implementations.

JMS provides two different messaging paradigms:

- **Point-to-point (PTP).** The destination in this paradigm is called a "queue." This domain allows for ***synchronous*** message delivery (see Figure 2).
- **Publish/subscribe.** The destination in this domain is called a "topic." This domain allows for ***asynchronous*** message delivery (see Figure 3).

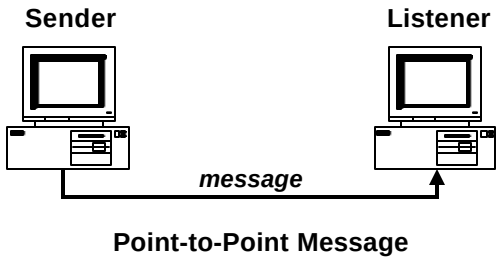

Figure 2. Point-to-point messaging

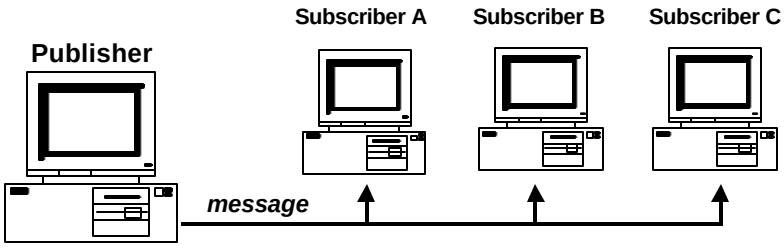

Figure 3. Publish/subscribe messaging

Table 1 lists interface names for each domain.

JMS Parent	PTP Domain	Pub/Sub Domain
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Destination	Queue	Topic
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReveiver, QueueBrowser	TopicSubscriber

Table 1. Interface names for JMS domains[1]

Extensible Markup Language (XML)

Like MOM, XML is helping the enterprise move away from tightly coupled computing environments. XML is an ISO-compliant subset of Standard Generalized Markup Language (SGML) and was approved as a meta language specification by the World Wide Web Consortium (W3C) in 1998.

XML provides a general syntax for describing hierarchical data. It is both machine-understandable and human-readable. It can be used to describe data within a wide range of contexts (document publishing, databases, e-Commerce, Java-based applications), as discussed below.

- **Documents and distributed processing.** XML is extensible in the sense that it is a meta language. In the context of documents, this means that a document developer (author) can write a Document Type Definition (DTD). A DTD is a domain-specific grammar written in XML for a specific type of document, such as a purchase order. The grammatical rules in a DTD in turn make it possible for the document receiver to interpret the document appropriately.

  For rendering documents/pages on the Web, client-side XML offers web designers more control and flexibility than using HTML. XML lets designers separate content from presentation. Style sheets can be created to describe presentation, and the client browser applies the style sheet to the XML document and renders the display.

  XML also lets developers use custom-designed tags for describing documents. Tags function as identifiers to signal the beginning and end of a related block of data. Using tags, developers can create a hierarchy of related data components. With document tagging, XML can help

[1] Sun Microsystems, "Java Message Service™," Version 1.0.1, October 5, 1998.

reduce the load on web servers because it is the browser client (not the server) that will take care of interpreting the tags.

- **Data and data exchange.** Server-side XML holds great promise as an open standard for B2B data exchange. For use in data description, XML tags can be used to specify information content rather than how information is to be displayed. XML organizes data hierarchically in a manner that can be completely specified. This lets XML parsers interpret the data appropriately.

  Data exchange can likewise be made easier with XML and DTDs. A validating XML parser can take a DTD and automatically check the syntax of a document and enforce business rules. The advantage is that developers no longer need to write customer application logic (extra code) for describing data. For example, enterprises can cut system administration costs by using XML-based technologies to describe and manipulate application configuration files.

XML, DOM, Middleware, and EAI

The Document Object Model (DOM) specification defines a programmatic interface for XML and HTML. This interface is platform- and language-neutral. It allows programs and scripts to dynamically access and update the content, structure, and style of documents. Vendors can support DOM as an interface to their proprietary data structures and APIs, and content authors can write to the standard DOM interfaces rather than product-specific APIs. These features allow developers to create standard interfaces for existing systems.

XML and DOM are emerging as EAI enablers because they can also be used to simplify data transformation and portability. Web integration servers can support EAI by extracting business data from systems and translating that data into XML. Once the data is in the normalized form of XML, it can be more easily interpreted.

Many ERP (PeopleSoft, SAP) and database (Oracle, IBM, and Sybase) vendors are providing XML interfaces that will promote data exchange in XML.

XML, Java, Messaging, and JMS

Since March 1999, efforts have been underway to develop a Java standard extension for XML. This extension will be a Java API. It will provide XML-specific features for developing XML-based services and applications.

XML is also an important part of the Java2 Platform, Enterprise Edition (J2EE). The J2EE includes XML as an enabler of B2B information exchange. For synchronous data messaging, Enterprise JavaBeans can be used to create a business service object. Related XML content can be sent using JavaServer Pages (JSP) technology. For asynchronous data messaging, JMS is the service of choice.

Java, Middleware, and EAI

The Java platform supports data exchange through a large set of middleware services (databases, transaction processing (TP) monitors, asynchronous messaging systems, ORBs). Because of Java's platform independence and the availability of middleware services, it is turning into a good environment for building EAI applications.

Furthermore, EAI often requires developers to access middleware services used in non-Java-based environments. XML can represent Java object data across different middleware services. The XML language thus can provide a data-centric method for getting disparate systems to interoperate. CORBA, on the other hand, uses a process-oriented method to achieve interoperability. But it is not always possible to use CORBA when a loosely coupled architecture is desired, and implementing CORBA may prove too complex for many installations. In these cases, XML can be used to describe the state of Java objects as they pass in and out of the Java Virtual Machine (JVM).

PROGRESS SONICMQ™

SonicMQ from Progress Software is a 100% Java-based messaging service that supports both point-to-point and publish/subscribe messaging paradigms. SonicMQ is one of the first full implementations of the JMS specification. The product not only complies with the latest version of JMS but goes beyond the minimum set of features called out in the JMS Specification (1.0.1, 10/5/98).

Most importantly, SonicMQ supports the XML standard data format. Progress has added other enhancements to the product to ensure high performance, reliability, flexibility, and ease of use.

- **High performance.** In multi-user environments, increased workload can degrade performance. SonicMQ offers multiple broker support and lets multiple servers be clustered. Workload can be shared across a server ***cluster***.

SonicMQ also features ***push technology*** to guarantee event notification to all subscribers. This eliminates the overhead incurred when subscribers continuously poll a repository for event information.

Asynchronous reply lets client subscribers continue processing while waiting for a reply. This capability is especially useful in business environments with mobile users who may be offline when messages are sent.

- **Reliability.** Reliability for a message service means ensuring that messages do in fact arrive at their intended destinations. SonicMQ offers built-in ***security*** (authorization, access control, digital certificates, encryption), ***fault tolerance*** through persistent messaging, and ***transactional support*** – all features that ensure data and message integrity.
- **Flexibility.** In addition to supporting both Point-to-Point and Publish/Subscribe messaging, SonicMQ extends the JMS message types to include ***XML***. This gives developers more options for address messaging. Developers can also customize message delivery: client applications can specify a particular Quality of Service (***QoS***), such as reliable/non-persistent or guaranteed/persistent.
- **Ease of use.** Usability is a key requirement for installing and maintaining any messaging system. SonicMQ offers both character-based and GUI-based ***administrative functions*** for monitoring message traffic.

  To promote ease of use in message addressing, SonicMQ supports ***subject-based*** as well as ***hierarchical*** namespace addressing methods. Messages can be addressed by subject or content. For the Publish/Subscribe paradigm, topics can be defined hierarchically. This allows messages to be sent to specific destinations, which can be either topics or subtopics.

  SonicMQ also frees up developers from having to manage the ***network infrastructure.*** The product takes care of socket and port management, protocols, semantics, and message transport mechanisms.

DIFFERENTIATORS

How does SonicMQ compare to other JMS implementations? First, it provides a compelling set of enhancements to ensure high performance and reliability. Secondly, it includes “native” XML support. (It ships with the IBM XML parser.) Third, costly add-ons are not needed to get a full, JMS-compliant messaging system up and running — the SonicMQ Developer Edition, with support for five clients on Windows NT, is free. Finally, Progress Software stands behind the

product: high performance and reliability have been the hallmarks of Progress product lines for many years.

THE PROGRESS ROAD MAP

Future Enhancements

Progress is committed to supporting SonicMQ and plans to add the following features in future releases:

- Additional platform support
- Interfaces to other messaging protocols, such as IBM's MQSeries, Microsoft Message Queue (also known as MSMQ)
- LDAP support
- Enhanced support for other clients (4GLs, C/C++)
- More support for XML
- Hot swap between brokers (failover and improved reliability)

Role of SonicMQ in Progress Software's Product Lines

SonicMQ is scheduled to be integrated into the next release of the Progress Apptivity Application Server (code-named "Vader"). Embedded in Vader, SonicMQ will provide the messaging infrastructure needed to support the EJB standard.

SonicMQ technology, including support for JMS and XML, is the cornerstone for bringing together the Progress Apptivity and 4GL product lines. By uniting these product lines, Progress is working toward its Universal Application Architecture (UAA), with support for Java, 4GL, COBOL, and C++ applications.

CASE STUDY: CHANNELINX.COM

ChanneLinx.com is a small, privately held ASP and ISV, headquartered in Greenville, SC. ChanneLinx is a B2B enabler. Its products include eLinx™, an electronic catalog that lets vendors present their products and services over the Internet; OrderLinx, which supports e-Commerce and integrates with vendors' back-end systems; and a new XML transaction processor and translator, called EIX.

ChanneLinx has used an early-release version of SonicMQ for the following purposes:

- Within an enterprise, as a queuing mechanism to exchange data (with the event-based publish/subscribe paradigm)
- Outside an enterprise, in a hosted environment, to exchange data in a supply chain
- In a peer-to-peer environment, when one application requests information from another application through the queue

According to Michael Quattlebaum, Director of R&D at ChanneLinx, the company chose SonicMQ because it needed to ensure reliability and cross-platform interoperability for messaging within its product line. Other products that ChanneLinx considered did not offer the same QoS that SonicMQ provides. Originally, the company planned to write its own JMS implementation but decided that SonicMQ fit its messaging requirements and supported XML as well. The combination of JMS and XML support made it unnecessary for ChanneLinx to bring in a third-party XML parser.

Prior to using SonicMQ, ChanneLinx only used the point-to-point (which they refer to as "peer-to-peer") messaging paradigm. By adopting SonicMQ, the company has broadened its support of messaging paradigms to include publish/subscribe. In new product releases of eLinx, the company will use publish/subscribe to publish an event to the message queue (for example, to announce the availability of a new document).

SonicMQ has delivered high performance, reliability, and ease of use to ChanneLinx. The company was able to get a proof-of-concept messaging system up and running within a day. Exposure to the SonicMQ's publish/subscribe support led them to change and improve the functionality of their flagship catalog product, eLinx.

CONCLUSION

What provides the foundation for success in today's e-Commerce marketplace?
Certainly, one of the building blocks is a computing infrastructure that supports fast and reliable data exchange. A standards-based, interoperable messaging service can provide the core for middleware services, both within an enterprise and B2B. But a message service on its own cannot ensure that disparate systems will work with each other — companies need to be sure that their data formats are compatible. MOM products are needed that can combine services like JMS with data format standards, such as XML.

For organizations that need to interconnect disparate systems for e-Business, Progress Software's SonicMQ product is one of the first JMS implementations that can meet this requirement.

PROGRESS SOFTWARE CORPORATION

14 Oak Park
Bedford, MA 01730
Phone: 800-477-6473
www.progress.com