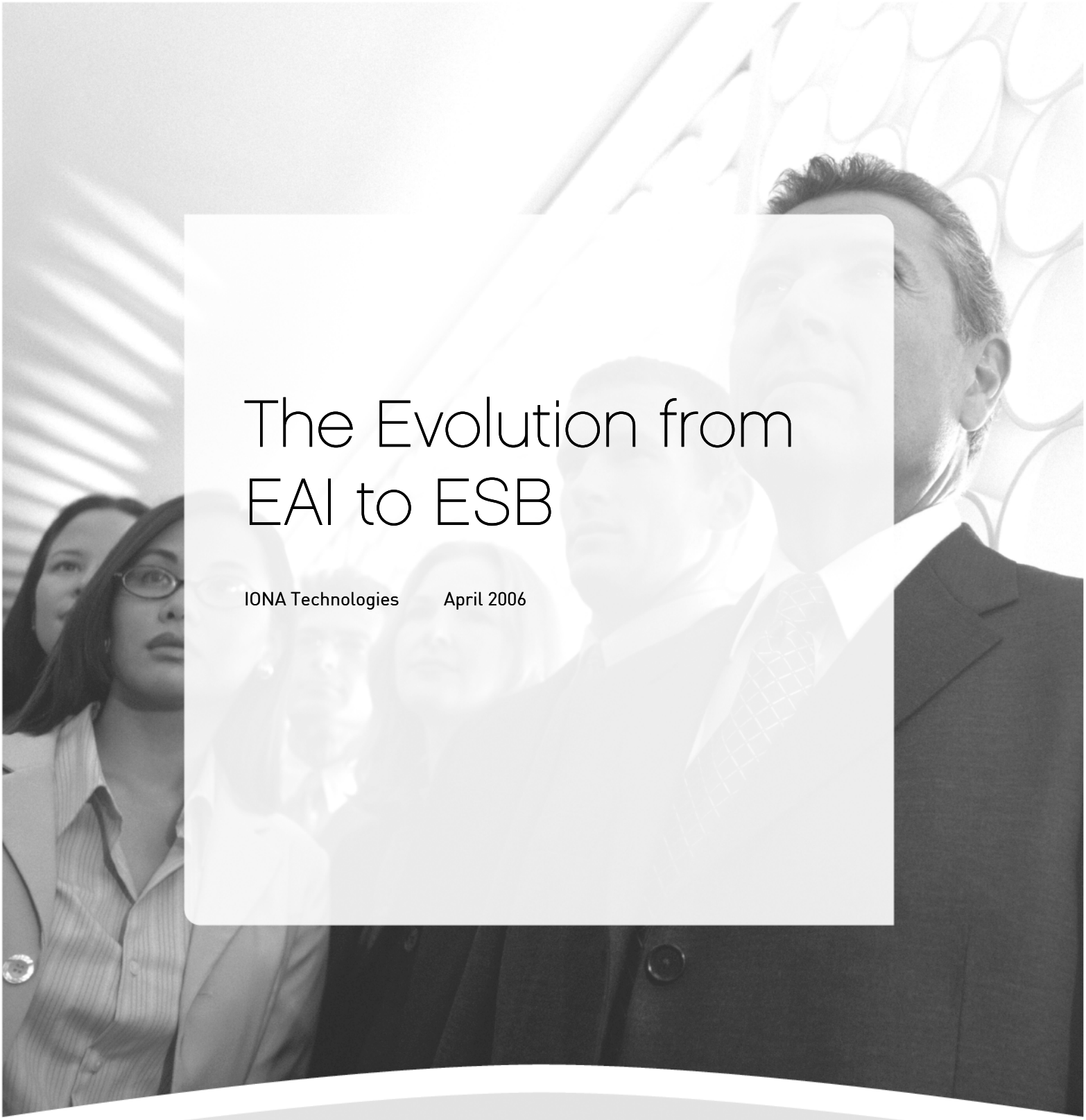

The Evolution from EAI to ESB

IONA Technologies April 2006

IONA

Making Software Work Together™

Introduction

As an industry leader, IONA is at the forefront of vision and production of enterprise integration technology. IONA's product line is constantly evolving in an ongoing effort to provide users with state-of-the-art enterprise integration capabilities. With the popularization of Service-oriented architecture (SOA) – an approach to system design, development and deployment that IONA's CORBA customers have practiced for years – a new category of integration products has emerged: the Enterprise Service Bus (ESB).

ESB is the software industry's name for the next generation of integration products. ESBs follow in the footsteps of Enterprise Application Integration (EAI) by adopting some of the more effective aspects of EAI technology while improving on EAI in other areas. Although the goals of EAI and ESB are the same, in the area of technical architecture the two products are quite different.

The Evolution of EAI

Historically, EAI technology was the software industry's first attempt to consolidate all of the disparate middleware solutions in the market into a single product suite. The need for EAI arose as companies sought to exchange information between separate silos of automation. Enterprise wide business initiatives such as customer relationship management (CRM) and enterprise resource planning (ERP) in the 1990's were the primary drivers for EAI systems.

Prior to EAI, the middleware landscape was dominated by an array of protocol stacks (such as CORBA, Tuxedo and MQ) and data formats (XML, XDR, Fixed, Variable, etc). Each of these technologies is largely capable of satisfying the integration needs of an enterprise on its own, but only if the selected protocol and data formats are ubiquitous in the enterprise. Unfortunately, the reality is that large and mid-sized IT shops are inevitably heterogeneous.

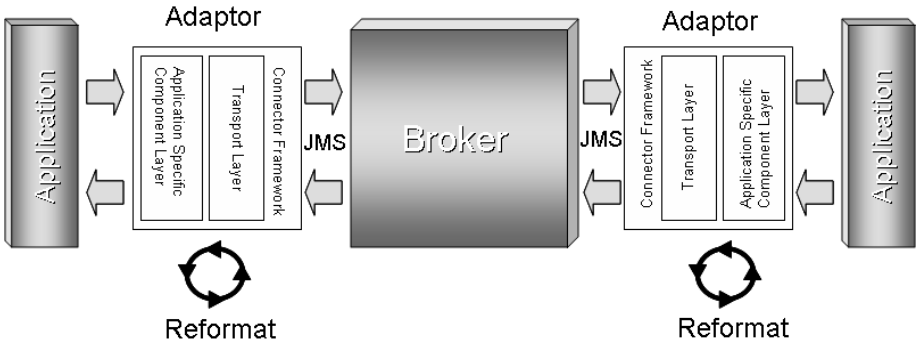

Figure 1: EAI broker acts as a hub

As shown in Figure 1, EAI took a simple, brute force approach to the problem of integrating dissimilar applications. EAI software created a hub that translates data and messages between different applications. The EAI hub used adaptors to reformat all incoming data into a common format called the canonical format that can be understood by the internals of the EAI hub as well as outgoing adaptors. Each adaptor was a substantial piece of software in its own right with multiple layers

managing application specific interactions and other transport layers managing connectivity to the application and the hub.

To implement a connection between EAI components, the EAI hub used an asynchronous message broker, like JMS, for all of its internal integration. In addition to reformatting the message payload, all interactions between applications went through multiple middleware transformations. Furthermore, the quality of services the application might require such as transactional processing and authentication/authorization security functions most often did not survive these transformations.

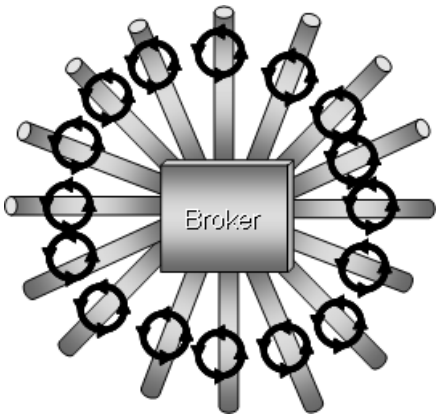

Figure 2: Hub constantly marshalling data

As a first generation, EAI was successful in that it provided a solution where none existed before. But there are inherent limitations in the EAI architecture that limit its ability to provide a sustainable enterprise-level solution. As shown in Figure 2, a central hub gives enterprises (or at least certain individuals within the enterprise) the advantage of centralized control. But the cost of constantly marshalling data into and out of these canonical formats creates an additional processing burden that requires the purchase of high-end servers and administrators to manage them. While most EAI solutions will allow you to deploy hubs in a cluster to get some additional scalability, this is only practical and performant to a limited degree and can quickly get quite expensive as you add more dedicated hardware.

EAI was vastly superior to hand coding every permutation of middleware and application interface in the enterprise application portfolio. It was a step forward that worked best when the mindset and the focus of the industry was on large-scale monolithic applications that needed to exchange data in support of these enterprise-wide initiatives. Since the first wave of EAI tools, vendors have tried to respond to the shortcomings of EAI in incremental ways. However, by continually adding new features, it made the EAI systems large, inflexible and hard to manage. Over the long term, a better technology was required if the vision of true enterprise integration was to be achieved.

The Evolution to ESB

The ESB is the next generation of enterprise integration technology, taking over where EAI leaves off. Like EAI, ESB is a technology that allows developers to integrate disparate systems that were created

using different middleware technology. ESB further improves on its EAI predecessor by adopting a more efficient and flexible internal architecture while leveraging the advantages of service orientation.

The connection between SOA and ESB is important to understand. SOA represents the policies, practices, and frameworks that enable application functionality to be provided and consumed as sets of services. As shown in Figure 3, a service is a business-complete logical unit of work, accessible programmatically from independently designed contexts via a direct openly documented interface. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface. Application software consists of services and service consumers (clients) in loosely coupled 1-to-1 relationships.

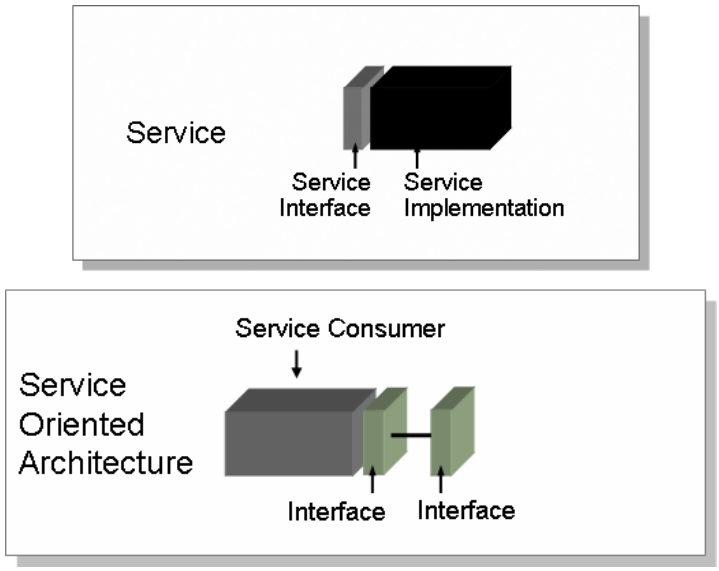

Figure 3: Services and SOA

SOA has been the software industry's response to the problem of managing large monolithic applications. As one might imagine, this difference in application architecture significantly impacts how application integration is best achieved. As shown in Figure 4, ESBs provide a backplane for integration among service providers and service consumers. New applications developed on modern platforms are inherently service-oriented. However, many existing enterprise applications are not designed with SOA in mind. In these cases, the ESB should provide the ability to expose these applications as services. Many of these EAI products are part of today's computing fabric and will continue to be used to solve integration problems, but for most cases moving forward the ESB has risen as a better alternative for the following reasons:

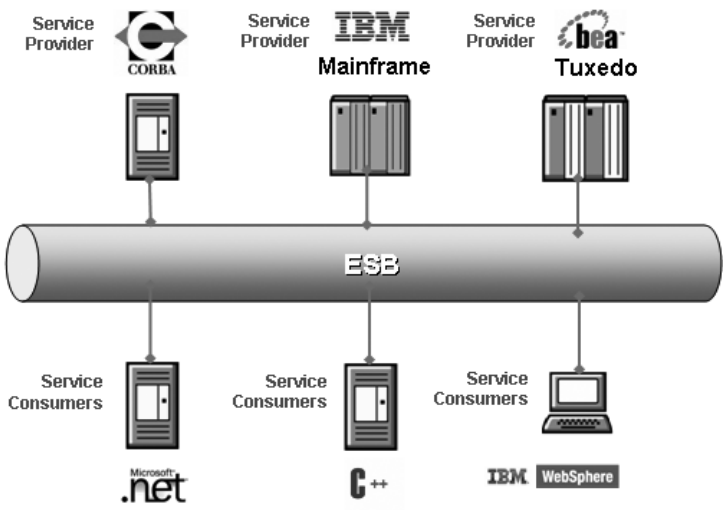

Figure 4: ESB provides a lightweight distributed architecture

Smarter endpoints - The ESB enables architectures in which more intelligence is placed at the point where the application interfaces with the outside world. The ESB allows each endpoint to present itself as a service using standards such as WSDL and obviates the need for a unique interface written for each application. Integration intelligence can be deployed natively on the end-points (clients and servers) themselves. Canonical formats are bypassed in favor of directly formatting the payload to the targeted format. This approach effectively removes much of the complexity inherent in EAI products.

Centralized versus distributed – Where EAI is a purely hub and spoke approach, ESB is a lightweight distributed architecture. A centralized hub made sense when each interaction among programs had to be converted to a canonical format. An ESB, like IONA's Artix, distributes much more of the processing logic to the end points. This is analogous to the difference between a mainframe and a modern distributed systems architecture. Hubs, like mainframes, can still be used where they make sense architecturally, but they are an option for the developer, not a vendor-mandated requirement.

No integration stacks – As customers used EAI products to solve more problems, each vendor added stacks of proprietary features wedded to the EAI product. Over time these integration stacks got monolithic and require deep expertise to use. ESBs, in contrast, are a relatively thin layer of software to which other processing layers can be applied using open standards. For example, if an ESB user wants to deploy a particular business process management tool, it can be easily integrated with the ESB using industry standard interfaces such as BPEL for coordinating business processes.

The immediate short-term advantage of the ESB approach is that it achieves the same overall effect as the EAI approach, but at a much lower total-cost-of-ownership. These savings are realized not only through reduced hardware and software expenses, but also via labor savings that are realized by using a framework that is distributed and flexible. In addition, an ESB can be deployed incrementally to reduce disruption and migration costs.

Artix - The Extensible ESB

IONA offers an ESB for distributed and flexible integration, that also provides owners of EAI systems an incremental solution to migrate to a SOA gradually.

The ESB is an architectural step forward, but some of today's ESB products lack all the capabilities that large enterprises require. The vast majority of these products depend on a single protocol (e.g., SOAP), programming language (e.g., Java), messaging transport (e.g., JMS), or deployment architecture (e.g., J2EE application server) that limits their ability to bridge multiple platforms. In addition, many ESBs are marketed by small startups with no experience in complex enterprise integration projects.

Artix is an extensible Enterprise Service Bus (ESB) that dramatically reduces operating costs for organizations with complex and heterogeneous IT systems by deploying, managing and securing a service-oriented architecture (SOA) without requiring a centralized hub. Artix uses distributed computing technology to leverage and modernize existing middleware investments to help the Global 2000 deliver products and services to their customers faster and more efficiently. Artix is

- **For incremental SOA adoption** – Artix creates a network of smart, standards-based endpoints using existing infrastructure so enterprises can begin with low-risk, high-value SOA projects and gradually add services as needed
- **Dynamic and adaptable** – Artix endpoints are independently configurable so services can be extended, modified and hot deployed without disrupting the rest of the enterprise
- **Technology-neutral** – Artix is a multi-platform and multi-protocol solution that connects diverse and lightweight endpoints without an expensive and cumbersome centralized server, and without promoting vendor lock-in

IONA has a proven track record of delivering mission-critical infrastructure, and has built many of the earliest and largest SOAs for Global 2000 customers, including Credit Suisse, BellSouth, Raymond James & Associates, Marconi, and Deutsche Post (DHL).

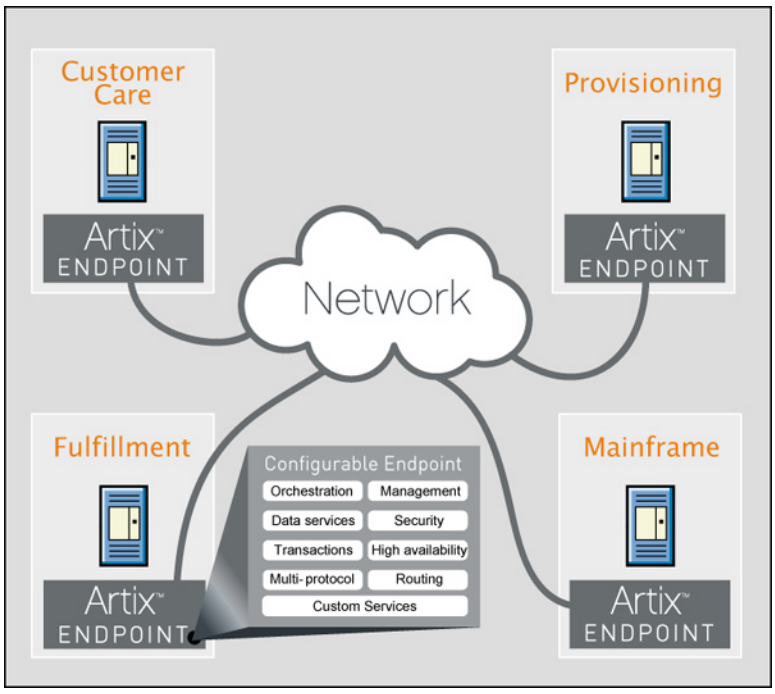

Figure 5: Artix distributed architecture

The Artix architecture, as shown in Figures 5 and 6, is based on a highly efficient microkernel and a series of plug-ins that allow it to adapt to any combination of middleware and application architecture. This includes adapting to new transport protocols, payload formats, security models, session management, transaction support and resiliency requirements. For integration systems, the Artix plug-in approach to extensibility has two important effects:

- Intricate or special purpose integration requirements can be addressed with new, specially designed plug-ins. This plug-in approach allows the integration runtime to stay small and fast, while addressing a broad range of integration issues.
- After the integration has been deployed, changes made to one of the application interfaces, or extensible endpoints, can be made without disrupting the service provided by that endpoint.

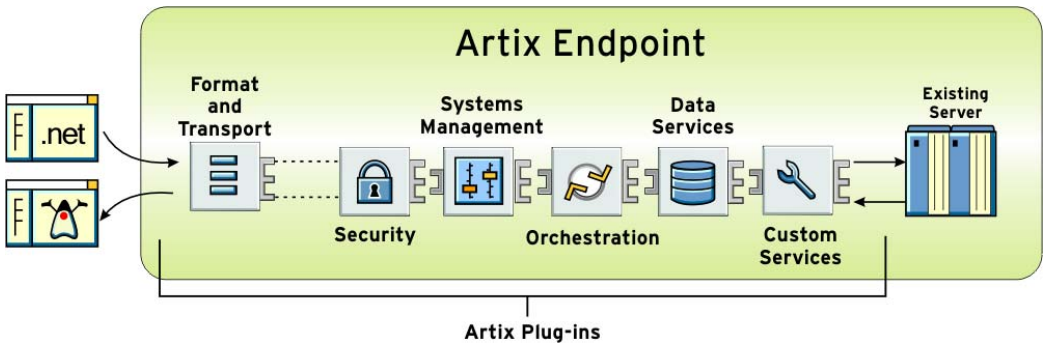

Figure 6. Artix Extensible Endpoint

Changes in the underlying architecture and infrastructure of the application are transparent to the other participants in the integration. Other features that further define the Artix product are its support for:

- High performance and small footprint integration with none of the traditional excess network data hops or transformation overhead
- Protocols and data formats of the dominant middleware and infrastructure - including the SOA platforms of IBM, BEA and Microsoft
- Standard application platforms of existing enterprise systems: mainframe transactions (IMS/CICS), C++ client/server apps and middleware platforms such as CORBA or BEA's Tuxedo
- Application styles of existing systems, including publish/subscribe, request/response, and asynchronous applications
- Emerging security standards of SOA platforms and Web services (WS-Security, Kerberos Tokens, etc.) and the legacy security models of existing enterprise systems
- Management via existing enterprise management infrastructures, including IBM Tivoli, CA Unicenter and HP OpenView
- Integration with run-time and development tools of the leading SOA platform suites

Artix users find that its extensibility allows them to leverage and extend their existing systems in place, without having to modify them. It allows the building of secure, fault-tolerant systems that are easier to manage and deploy. And the product's extensibility promotes a small footprint, very high performance integration solution, which in turn meets enterprise throughput requirements with significantly reduced overhead.

Summary

As with the previous evolution from hand coding to EAI technologies, the evolution of middleware from EAI to ESB is putting greater power and flexibility in the hands of developers and users while reducing the overall costs of building systems. In many ways, the evolution from EAI to ESB has the same motivations as the evolution from monolithic enterprise applications to service-oriented applications. In both cases, users have hit a wall of complexity in which a simpler more flexible approach is essential. IONA is committed to providing the industry with both the thought leadership and product leadership this evolution requires.

Global 2000 companies, with their heterogeneous systems and application silos, are using Artix to reduce the complexity within their enterprises. IONA, with its history of delivering high performance, mission-critical applications for over ten years is helping these companies repurpose their current investments and incrementally move towards an open architecture.

IONA customers - including BellSouth, AT&T, Marconi, and DHL (Deutsche Post) – use an extensible ESB to service-enable existing systems and establish a scalable, adaptable, SOA architecture. They chose Artix for its broad platform support, extensibility, and enterprise quality of service, and today they are building new business applications and process flows on common platforms such as Microsoft's .NET Framework, IBM's WebSphere, or BEA WebLogic.

In looking to the future, IONA's customers can count on IONA as a business partner that will provide them with the best the industry has to offer, today and tomorrow.

IONA Technologies PLC
The IONA Building
Shelbourne Road
Dublin 4
Ireland
Phone +353 1 637 2000
Fax +353 1 637 2888
Support: support@iona.com
WWW: www.iona.com

IONA Technologies Inc.
200 West Street
Waltham MA 02451
USA

Phone +1 781 902 8000
Fax +1 781 902 8001
Training: training@iona.com

IONA Technologies Japan Ltd
Akasaka Sanchome Building 7/F
3-21-16 Akasaka
Minato-ku Tokyo
Japan
Phone +813 3560 5611
Fax +813 3560 5612
Sales: sales@iona.com

Making Software Work Together™