Fiorano

Enabling change at the speed of thought

www.fiorano.com

ESB Competitive Analysis

A Guide to selecting the right Enterprise Service Bus

AMERICA'S
Fiorano Software, Inc.
718 University Avenue Suite 212, Los Gatos,
CA 95032 USA
Tel: +1 408 354 3210
Fax: +1 408 354 0846
Toll-Free: +1 800 663 3621
Email: info@fiorano.com

EMEA
Fiorano Software Ltd.
3000 Hillswood Drive Hillswood Business Park Chertsey Surrey
KT16 0RS UK
Tel: +44 (0) 1932 895005
Fax: +44 (0) 1932 325413
Email: info_uk@fiorano.com

APAC
Fiorano Software Pte. Ltd.
Level 42, Suntec Tower Three 8 Temasek Boulevard 038988
Singapore
Tel: +65 68292234
Fax: +65 68292235
Email: info_asiapac@fiorano.com

Table of Contents

Executive Summary

Traditionally Enterprise Architects have had two choices for business integration:

1. Implement custom point to point solutions for each project as they come up, leading to a rigid architecture with a high cost of maintenance or
2. Buy a complex, expensive and proprietary integration broker suite which requires a large up-front cost, long implementation schedules and a substantial investment in consulting services to maintain.

Now they have a third viable choice, an ESB (Enterprise Service Bus) - a neutral, standards-based integration solution that is flexible, robust, scalable and at the same time easy to implement and maintain. The power of an ESB to solve an Enterprise's integration problems has been discussed by several leading Technology Analysts such as Gartner, IDC, Saint Consulting and others.

"ESB offer an excellent basis for integration, delivering a flexible and adaptable environment that enables integration projects to be put in place productively, effectively and in a staged manner" says Steve Craggs, VP of the EAI Industry Consortium.

"An ESB can be a sensible first step toward a systematic Enterprise Nervous System because it provides the basic connectivity backbone. It can interoperate with a variety of disparate application servers simultaneously, smoothing over technical differences and also providing services for communication and integration" adds Roy Schulte, Vice President Research at Gartner Inc.

To fully realize the potential of its investments in IT infrastructure it is therefore essential that an Enterprise carefully evaluate its choice of an ESB that meets its current needs and also has the flexibility to extend and scale with a growing and changing enterprise. To assist such an evaluation, this whitepaper compares currently available ESB's based on key characteristics as described by Steve Craggs, Vice Chairman of the EAI Industry Consortium in his White Paper on "Best-of-Breed ESB's".
Read the complete whitepaper at: http://www.eaiindustry.org/docs/BestofBreed_ESBs.pdf.

The technical comparison was aimed at providing an in-depth investigation of the design, core servers, tools and features that comprise the current requirements for successful business integration. The results of the analysis are based on an extensive hands-on product review of multiple ESB implementations. This document provides a comprehensive summary of the findings.

Detailed Technical Comparison

The tables below provide a detailed technical comparison between the Fiorano Enterprise Service Bus and other ESB's based on these identified characteristics.

Fiorano
Enabling change at the speed of thought™

Fundamental ESB Services

Key Characteristics	Fiorano ESB	Other ESB's
SOA Implementation	❖ Flexible and extensible Service abstraction model, allowing multiple inputs and outputs ❖ Allows complex integrations to be composed or 'assembled' from pre-built, pre-tested Services ❖ Packaged support for EJB, COM, Web Services and CORBA components ❖ Packaged support for multi-language services including: C, C++, C#, and Java executables, scripting and legacy services	❖ Control-flow semantics limit the Service abstraction model; no model for coarse-grained "agent-like" services that can be dynamically deployed ❖ Reasonable support for EJB and Web services ❖ Typically tied to a single technology (such as .NET or Java); limited support for multi-language services
Standards-Support	❖ Embedded, High-Performance JMS messaging with multi-language support ❖ Leverages standards for connectivity (JCA), Transformation (XSLT) and security (SSL) and Web Services (SOAP, UDDI)	❖ Non-pluggable, vendor specific, tightly coupled messaging; often non-standard; limited multi-language support ❖ Leverage standards including JCA, XSLT and J2EE; typically biased towards Java Standards
Messaging **XML Support** **Transformation** **Intelligent Routing** **Communication Services**	❖ XML message support, with integrated Schema validation Integrated transformation and mapping tools, with support for XSLT, XPATH and Content Based Routing ❖ Loosely coupled, "external" routing for dynamic, on-the-fly changes to application flows ❖ Peer to Peer JMS-compliant transport	❖ XML message support with schema validation ❖ Integrated transformation tools; some ESBs have poor support for XML ❖ Tightly-coupled routing typically tied into message headers, inhibiting dynamic, runtime changeability of flows ❖ Hub-and-Spoke messaging, limiting scalability and adding to hardware costs at the hub
Highly Distributed Implementations **Service-based Approach** **Location and Technology Transparency** **Intelligent Routing** **Single Point of Control** **Deployment Support**	❖ Fully distributed service-based architecture with centralized control and distributed, parallel data-flows between services; Peer Messaging Servers enable parallel data-flows between network end-points, enhancing scalability and performance ❖ Inter-Service communication over JMS supported with location transparency ❖ Dynamic "external" data routing between services, allowing data-routing changes between running services; Content and message-header based routing supported	❖ Limited distributed architecture: data flow between services is always via the centralized Message Hub, decreasing performance and scalability. ❖ Support for location transparency ❖ Limited support for flexible, on-the-fly routing changes between running distributed services; service-instance-based routing typically not supported; content based routing typically supported. ❖ Limited centralized control of applications and services; sparse support for monitoring, versioning and dynamic tracing

	❖ Centralized Control for application monitoring, document tracking, deployment, versioning, service updates/upgrades, dynamic tracing and logging ❖ Dynamic deployment of Services from centralized control point to network end-points without restarting flows/applications.	❖ No support for remote deployment or automatic upgrades of services; typically, services need to be manually updated and restarted for each service upgrade/update.
Connectivity **Web Services** **J2EE Connectors** **JMS** **WebSphereMQ**	❖ Pre-packaged connectors for Web-Services, J2EE (EJB), JMS, MQSeries (WebSphere MQ)	❖ Pre-packaged connectors for Web-Services, J2EE (EJB) and JMS and other enterprise middleware
Administration **Single Point of Control**	❖ Centralized administration of entire ESB infrastructure network with support for Support for HP-OpenView, IBM Tivoli and other SNMP-compliant network management tools	❖ Centralized remote administration supported ❖ Some ESBs have limited support for SNMP-compliant network management tools
Deployment **Dynamic Change**	❖ Services can be configured, upgraded and automatically deployed to any node in the network using integrated tools from a central point ❖ Service versioning and a rule-based configuration engine allows fine-grained control of service deployments to network end-points ❖ Dynamic run-time deployment allows services to be added or replaced on-the-fly within running applications for incremental extensibility ❖ Sub-flows allow complete applications to be 'packaged' into single services, enhancing reusability	❖ Typically have no or limited support for automatic ❖ deployment of services on remote nodes ❖ Remote service containers need to be manually configured at network end-points for each service update or change ❖ Limited support for versioning of services or configuration management of services ❖ Service-aggregation limited by "BPM-like" control-flow model ❖ Applications containing multiple service instances cannot easily be packaged into single components, limiting reusability
Monitoring **Problem Determination** **Problem Prediction** **Internal Support** **Support for Enterprise Management Frameworks**	❖ Problem determination via real-time real-time alerts for service-operations (running, stopped, service-specific messages); comprehensive monitoring API for services; easy adjustment of required alerts at runtime using point and click tools ❖ In-built alerts for common system problems: disk-full, thread and process limits, etc. Additional problem prediction via service level	❖ Limited support for monitoring and data tracking; limited support to adjust definitions of required alerts at runtime ❖ No support for data debugging between remotely running services ❖ Limited support for SNMP or other industry-standard Enterprise Management frameworks; some ESBs incorporated support for JMX-based administration

	log-tracking, fault and error flows, and dynamically insertable distributed data-debugging capabilities ❖ Internal capture of comprehensive system statistics ❖ In-built support for SNMP and other Enterprise Management Framework Agents like OpenView and Tivoli, for comprehensive reporting	
System Actions **Single Point of Control** **Remote access capability** **Start/Stop Facilities** **Manual Routing Support** **Tracing** **Message Editing**	❖ Single point of control for distributed applications; composition, web-based monitoring, launching, security and administration tools centralized on a per-domain basis. ❖ Remote start/stop supported at both on a service-instance as well as application levels ❖ Manual routing supported without additional programming via GUI tools ❖ Dynamic service-level tracing allows fine-grained, runtime adjustment of logging information ❖ Message-Editing and modification via distributed debugging tools, with support for dynamic break-points between message flows	❖ Single point of control via central administration tools; limited support for monitoring and security tools ❖ Remote Start/Stop limited to complete applications; not available at Service-instance level ❖ Manual routing typically requires additional programming and knowledge of low-level messaging details ❖ No support for tracing and dynamic logging of services ❖ Limited support for message editing; control-flow semantic model severely limits the ability to debug data-flows at runtime

Robustness

Key Characteristics	Fiorano ESB	Other ESB's
Fault Avoidance **Standards Adoption** **Ease of Use** **Scalability**	❖ Multiple standards supported, including XML, JMS, J2EE, SOAP/UDDI, and .NET, with development in all major languages ❖ Easy-to-use, intuitive GUI tools for composing, deploying, monitoring and administering distributed applications; no programming required to set up and deploy complex distributed processes using pre-built, pre-tested services ❖ Enhanced fault avoidance via Sub-flows and error-flows embedded within the application-process flow ❖ Dynamic addition of Peer Servers at network end-points for improved scalability and	❖ Standards support includes XML, JMS, J2EE, SOAP/UDDI, but development typically biased towards a single language (Java, .NET, etc.). ❖ GUI Tools to compose distributed applications based on a control-flow model; manual intervention and middleware knowledge/programming typically required to deploy applications ❖ Limited support for GUI tools for fault-handling; manual programming required on a per instance basis to handle error conditions ❖ Tight binding of Services to Service-

	performance	containers requires significant time to deploy services at network end-points
Fault Tolerance **Routing around failure** **Redundancy Support** **Recovery**	❖ Dynamic message re-routing across services avoids route failures; store-and-forward at local peers ensures data integrity in case of unavailable paths ❖ Redundant peer-labels for Services ensure automatic fire-up for backup service instances on different machines in case of primary node failures ❖ Clustered Enterprise Server with real-time mirroring of application state ensures complete fault-tolerance ❖ Compensating transaction model for recovery and long-running transactions	❖ Centralized messaging, directory and administration servers with limited clustering support, leading to single points of failure ❖ No concept of distributed data-flows between network end-points (due to inherent control-flow semantic models), making data re-routing impossible ❖ No support for service level failover ❖ Service crashes can result in data loss; remote restart of services unavailable, complicating recovery

Scalability and Performance

Key Characteristics	Fiorano ESB	Other ESB's
Performance **Asynchronous Messaging** **Multi-Threading** **Load balancing** **Large data handling**	❖ High-performance JMS backbone with support for asynchronous messaging and multi-threaded services; Messaging servers at network-end points enable direct, parallel data-flows between distributed services, allowing unmatched performance and scalability ❖ Load balancing supported at the Service level, allowing dynamic load-balancing to be added to running applications ❖ Available Services for bulk data transfer (gigabyte file-transfers)	❖ Messaging-backbone supports async messaging and multi-threaded services; centralized messaging limits performance and scalability ❖ No Service-level load-balancing ❖ Limited support for large sized messages
Scalability **Prioritization Services** **Classes of Service** **Dynamic change support** **Transparent resource addition**	❖ Parallel data-flows between distributed services for linear, unbounded scalability; no centralized bottleneck to communication ❖ In-built service-categorization and message prioritization on data-flow routes ❖ In-built dynamic modification and extension of applications at runtime, with control over individual service instances	❖ No support for parallel data-exchanges between distributed services; all data flows through centralized hub, which becomes a bottleneck to communication ❖ Limited to No support for service categories. ❖ Limited support for dynamic application extension (due to limited control-flow semantic models); no control over individual service instances; a single slow

	❖ Dynamic changes to a Multi-level Service categorization to ease management of service groups ❖ Incremental, transparent resource addition at both infrastructure as well as Application and Service levels; infrastructure can be extended incrementally at runtime, without disrupting running applications and services	service can slow down an entire business process with no easy fix/recourse due to control-flow semantics. ❖ Utilization of transparently added resources (infrastructure and services) requires intrusive manual modification of applications

Security

Key Characteristics	Fiorano ESB	Other ESB's
Access Control **User Authentication** **Component Authorization** **Non-Reputation**	❖ ACL-Based, J2EE and LDAP compliant security for User and Service authentication ❖ In-built support for sophisticated "configuration management" for fine-grained runtime control over Service and Application deployments ❖ Service-level digital signatures ❖ Non-repudiation enforced via document tracking and audit-trails controllable at service-instance level	❖ ACL-based security with service authentication ❖ Limited to no support for "Configuration Management" ❖ Limited document tracking support
Information Security **Privacy(encryption)** **Integrity checking**	❖ Pluggable RSA and DSA encryption on all data-flow routes ❖ SSL-based transport level security at transport (JMS) level	❖ Pluggable encryption of data flows typically requires manual intervention/setup ❖ SSL-based security at transport level
Tools Usage **Authorized users only**	❖ Role-based security model controls access to and usage of tools on a per-user basis; multiple groups with user-rights assignment	❖ Some ESBs provide only limited support for role-based security

Breadth of Connectivity

Key Characteristics	Fiorano ESB	Other ESB's
Legacy Systems	❖ Third-party support for mainframe and legacy adapters ❖ Native pre-built SAP adapter	❖ Third-party support for mainframe and legacy adapters
Other EAI Solutions	❖ Pre-built, native bi-directional bridges to JMS, MQSeries, Tib/RV and MSMQ	❖ Many ESBs provide MOM/Broker adapters via third party support
Application Servers **WebSphere, Weblogic, Others**	❖ Native EJB adapter allows asynchronous invocation of WebServices and EJBs hosted on any J2EE compliant AppServer.	❖ Native EJB adapter allows asynchronous invocation of WebServices and EJBs hosted on any J2EE compliant AppServer.

Other Standards **.Net** **COM/CORBA**	❖ Native .NET integration via C#, Visual Basic and ActiveX support ❖ Native support for Serviced developed in Java, C, C++, Perl and other scripting languages ❖ Native COM and CORBA components for Service development ❖ Support for Web Services and JCA	❖ Limited support for .NET ❖ Limited support for COM or CORBA ❖ Restricted scripting (Java Script) ❖ Support for Web Services and JCA
Internet Facilities	❖ Native Services for HTTP connectivity and FTP	❖ Many ESBs bundle Services for HTTP or FTP

Tools

Key Characteristics	Fiorano ESB	Other ESB's
Configuration **ESB definition tools** **Flow control definitions** **GUI interface** **Ease of use**	❖ Sophisticated, easy to use GUI tools allow developers to easily create data flow definitions across distributed services with drag-drop-connect metaphor ❖ Easy-to-change and modify workflows and data-flows at runtime using graphical tools	❖ Data-flow and event-flow setup across distributed services requires knowledge of messaging structures such as Queues and Topics ❖ Control-flow model inhibits ability to modify data-flows at runtime; applications must typically be stopped, modified and redeployed
Connectivity **Wrappers**	❖ Sophisticated 'wrapping' API; rich, pre-built, pre-tested components including: ❖ Relational Databases, with XML mappings to/from the database ❖ Sophisticated File Adapters with File Schema Editors that allow complex XML targets for flat-file structures ❖ Adapters to IBM Websphere MQ (MQSeries), MSMQ, JMS messaging systems ❖ Web Services (Componentized WSDL), EJB, SAP, PeopleSoft, and ❖ Utility Adapters including SMTP, POP3, HTTP, FTP and more	❖ Pre-built Services support varies between ESBs; many ESBs provide support for the basic adapters such as Database, File, HTTP ❖ Some ESBs have limied support for XML generation from flat files
Incremental Deployment **Wrappers** **Location/Technology transparency** **Service-based approach** **Manageability**	❖ Wrapping API and Component model allows existing legacy applications to be incorporated as Services in a non-intrusive manner ❖ Location transparency built into component model; Services can execute anywhere since routing information is detached from Service code ❖ Tools for managing multiple versions of components, allowing version upgrades to be driven from	❖ Service creation supported in a control-flow model; dynamic "agent-like" deployment to remote end-points typically not supported, leading to increased costs of deployment ❖ Location transparency is supported by JMS. Service development support is typically biased towards single language ❖ Limited to no support for component versioning

	a centralized location via GUI tools	
Life Cycle Support **Development, Test, QA, Production**	❖ Component state labeling (Development, QA, Staging, Production), combined with the labeling of network end-points give deployment engineers control over the classes of components running on particular hardware systems within the network, allowing unparalleled deployment flexibility unmatched by any other platform	❖ No concept of Component State labeling or Deployment Management ❖ Limited to no support for managing multiple versions of services or components

About Fiorano Software

Fiorano Software (www.fiorano.com) is a leading provider of enterprise class business process integration and messaging infrastructure technology. Fiorano's network-centric solutions set a new paradigm in ROI, performance, interoperability and scalability. Global leaders including Fortune 500 companies such as Boeing, British Telecom, Credit Agricole Titres, Lockheed Martin, NASA, POSCO, Qwest Communications, Schlumberger and Vodafone among others have used Fiorano technology to deploy their enterprise nervous systems.