Fiorano

Enabling change at the speed of thought

www.fiorano.com

Fiorano ESB Architecture

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

Chapter 1: Introduction

Emerging standards for enterprise communication, connectivity, transformation, portability, and security have tried to simplify the enterprise integration and middleware problem. The Enterprise Service Bus (ESB), a new variation of software infrastructure, has added to the range of standards-based technologies enterprises can use as the Enterprise Nervous System (ENS) backbone. This chapter provides an overview of the Fiorano ESB from Fiorano.

Today's enterprise networks typically deploy hundreds of applications from different vendors. There is little standardization of communication protocols between individual enterprise systems, and exchanging data between applications from different vendors is surprisingly hard. The lack of a standard platform for distributed enterprise applications increases the cost and complexity of developing and deploying business solutions. The Fiorano ESB is a new breed of enterprise middleware designed to alleviate these and other problems.

The concepts underlying the Fiorano ESB architecture and implementation are not revolutionary. They have evolved with the gradual emergence of enterprise standards for communication, connectivity, transformation, service-oriented application construction, portability, and security. The Fiorano ESB promises, for the first time, the creation of a true standard enterprise backbone for deploying business processes, collaborative systems, and distributed business solutions.

1.1 Enterprise Solution Requirements

Requirements for enterprise solutions have been intact for 20 years. Enterprise solutions typically consist of one or more distinct applications (also known as services or components), combined into a single distributed solution with the following characteristics.

- **Communication:** Services need to communicate reliably with each other over the network. A reliable, scalable, robust, and location-independent communications system dramatically reduces the development time for distributed systems while increasing reliability.
- **Connectivity:** To extract data from a service, one first needs to be able to easily connect to that service. Due to absence of any standards, this has been difficult.
- **Transformation** Data produced by a given service is typically not easily understood by another service. To make the data digestible by another service, it first needs to be suitably transformed.
- **Service-Oriented Application Architecture:** Distributed enterprise systems span multiple nodes and operating systems, and typically consist of several independently running enterprise services loosely bound to each other via event-driven messages. This SOA for application composition allows incremental, dynamic extensibility and greatly reduces costs of maintenance and Total Cost of Ownership (TCO).
- **Portability:** Most enterprises have a variety of computer systems, ranging from thin-clients and Windows desktop PCs to higher-end UNIX servers and mainframe systems. Portability and ease of communication between different operating environments remain concerns for enterprise solutions.

- **Security:** Finally, all connectivity to and communication between enterprise services need to be secure at levels satisfactory to the enterprise. Since distributed applications span different departments and locations within and outside the firewall, security is highly important.

1.2 Early Integration Solutions

Several companies developed Enterprise Application Integration (EAI) solutions during the mid-'90s with the above design patterns in mind. Unfortunately, all these solutions were proprietary. WebSphere MQ and TIBCO Rendezvous are good examples of proprietary communications buses. Several companies developed proprietary connectors to many packaged enterprise applications (including SAP, PeopleSoft, and Oracle Applications, among others). Yet, lacking any standards for connectivity, brand new adapters had to be developed for different versions of each enterprise system or application. This resulted in tremendous maintenance problems.

Without standards for transformation, each EAI software vendor also developed customized transformation engines, leading to even more proprietary middleware. Most solutions were developed in C/C++ and other proprietary, non-portable languages, resulting in vastly increased porting and maintenance costs. Finally, security was typically added as an afterthought to most EAI and enterprise middleware platforms, resulting in a fragmented security model.

It's no wonder CIOs have felt they had a difficult set of problems on their hands.

1.3 Standards Emerge

Emerging standards for enterprise communication, connectivity, transformation, portability, and security have greatly simplified the enterprise middleware problem. Enterprises should, at all costs, avoid dependence on a single vendor. This is possible now because of the opportunity to apply truly open industry standards in a heterogeneous, best-of-breed environment. Standards-based technology coordination expands choices and is less costly. Specific standards include:

- **Communication Standards:** Since 1998, the Java Message Service (JMS) has emerged as the dominant industry standard for enterprise communication, implemented by thousands of companies. JMS has gained such a strong mindshare that other Message- Oriented Middleware (MOM), such as IBM's WebSphere MQ, has had to offer a JMS implementation. The secret to the popularity of JMS is that it combines a level of functionality suitable for almost all market needs with an attractive price point and the benefits of being an industry standard.
- **Connectivity Standards:** Web services standards allow any enterprise system or application to efficiently expose interfaces to the external world, dramatically simplifying the problem of connectivity to enterprise systems. These standards include Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), and Web Services Description Language (WSDL). SOAP provides Remote Procedure Call (RPC) capabilities for XML. UDDI provides a searchable registry of XML Web services. WSDL is an XML-based Interface Description Language (IDL) for describing XML Web services. XML is the standard format for Web data, and is beginning to be used as a common data format at all levels of the architecture.

- **Transformation Standards:** Over the years, eXtensible Stylesheet Language Transformation (XSLT) and Xquery have emerged as enterprise standards for transformation, with support from most vendors and widespread acceptance from customers. XSLT transforms XML documents from one schema into another; it's used for data interchange between systems using different XML schema, or mapping XML to different output devices.
- **Service-Oriented Architecture (SOA):** The emergence of SOAs lets you compose complex distributed applications (spanning multiple platforms and operating systems) as a set of Enterprise Services with a defined form of invocation, both asynchronous and synchronous. This approach allows the composition or assembly of applications from prebuilt, pre-tested services; the composed application is easy to modify or extend via service addition or replacement. This directly promotes reuse within the enterprise, decreasing time-to-market and system TCO.
- **Portability:** The Java programming language (used today by more than three million programmers worldwide and adopted by all Fortune 1000 companies) is the standard for building portable enterprise applications and middleware. Modern middleware technologies implemented in Java run unchanged across multiple hardware and operating systems, including Windows, UNIX, mainframes, and midsize systems.
- **Security:** The widespread acceptance of J2EE- and LDAP-compliant security gives administrators fine-grained control over the execution of applications and services on machines across the network. Additionally, SSL-based transport level security based on Secure Sockets Layer (SSL) provides robust security mechanisms for privacy and integrity checking. Lightweight Directory Access Protocol (LDAP) is a subset of X.500 designed to run directly over the TCP/IP stack. LDAP is, like X.500, both an information model and a protocol for querying and manipulating it. LDAPv3 is an update developed in the Internet Engineering Task Force (IETF), which addresses the limitations found during deployment of the previous version of LDAP. SSL is an open, non-proprietary protocol for securing data communications across computer networks. SSL is sandwiched between the application protocol and connection protocol. SSL provides server authentication, message integrity, data encryption, and optional client authentication for TCP/IP connections.

The presence of these foundation standards has spurred the evolution of a standards-based enterprise backbone: the Fiorano ESB.

1.4 The Fiorano ESB

The Fiorano ESB is an enterprise platform that implements standardized interfaces for communication, connectivity, transformation, portability, and security.

The Fiorano ESB implementation implements:

- Standards-based communication infrastructure (for example, JMS)
- Standards-based connectivity including Web services, Java 2 Enterprise Edition (J2EE), and .NET adapters. (Sun's J2EE and Microsoft's .NET are the two dominant distributed computing architecture frameworks. J2EE provides portability of a single language [Java] over multiple operating systems and hardware platforms. .NET supports a wide range of languages but is primarily tied to the Microsoft Windows operating system and Intel hardware.).
- Standards-based transformation engines (for example, XSLT and Xquery).

- SOA for application composition and deployment
- Standards-based security (for example, LDAP and SSL

The Fiorano ESB implementation supports development in multiple programming languages. This, coupled with the inherent portability of the underlying infrastructure, makes the Fiorano ESB a true multi-language, multi-platform enterprise backbone.

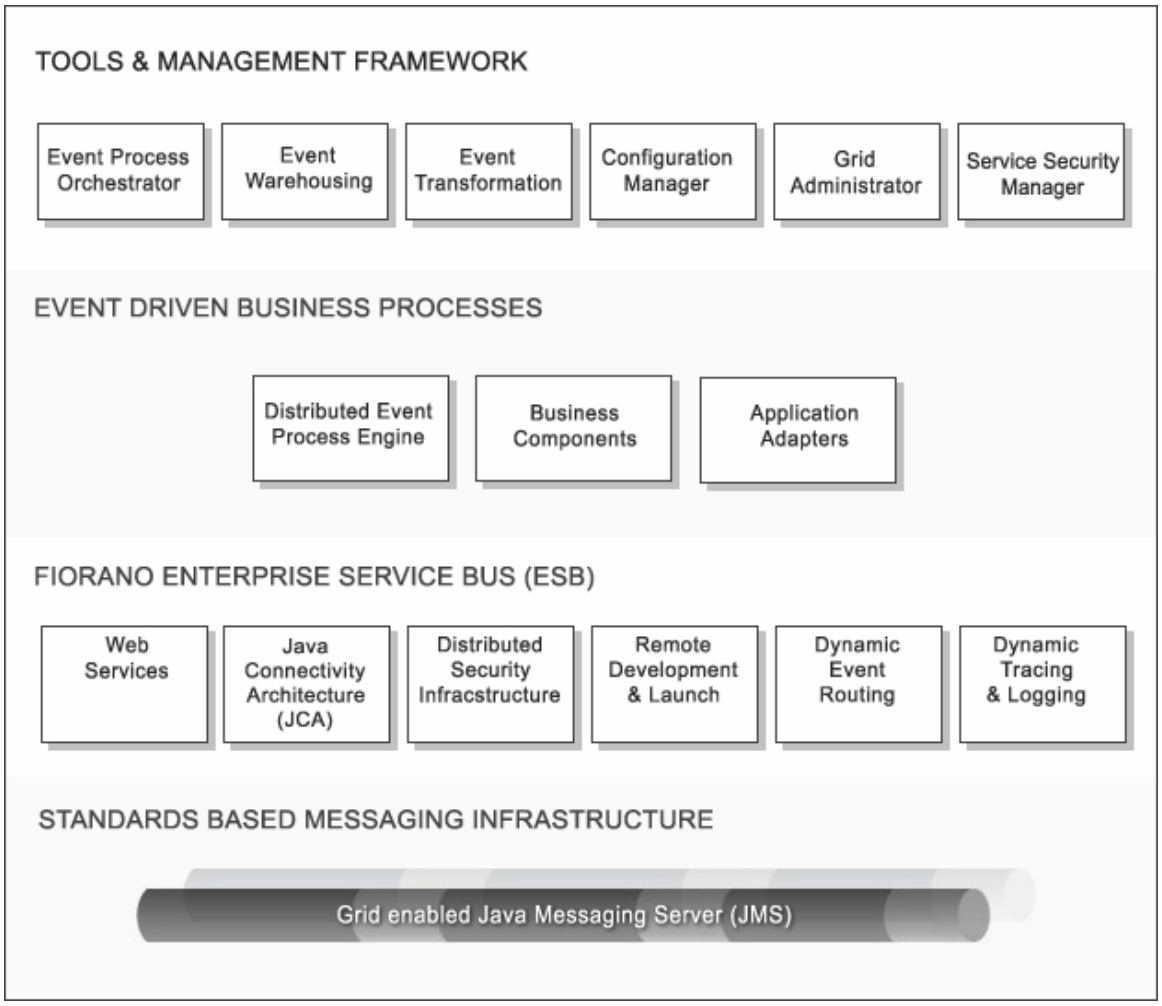

Figure 1.1: ESB Architecture

1.5 Fiorano ESB Benefits

The Fiorano ESB leverages recent integration technology enhancements into a standards-based, affordable package. Fiorano offers several advantages over existing, proprietary integration solutions:

- **Extended, Standards-based Connectivity:** The Fiorano ESB incorporates a standards-based JMS-compliant messaging backbone, letting systems within and across the entire value chain easily exchange information via asynchronous or synchronous messaging. ESBs provide enhanced systems connectivity using Web services, J2EE, .NET, and other standards.
- **Flexible, Service-based Application Composition:** Based on SOA, the Fiorano ESB application model allows complex distributed systems, including integration solutions spanning multiple applications, systems, and firewalls, to be composed from pre-built, pre-tested services. This provides easy extensibility.
- **Reduced TCO via Enhanced Reuse:** The SOA approach to application composition directly promotes reuse, easing maintenance and further reducing system TCO.
- **Reduced Time-to-market and Increased Productivity:** The Fiorano ESB provides these benefits through the reuse of components and services, and the ease of application composition offered by an SOA, standards-based communication, transformation, and connectivity.

All these benefits derive from the strong support for standards in each component of the Fiorano ESB architecture: communications, connectivity, transformation, portability, and security.

1.6 Fiorano ESB Technical Advantages

The Fiorano ESB provides significant added value in comparison to existing integration solutions. Some of the core characteristics of Fiorano ESB include:

1.6.1 Robustness, Scalability and Performance

The Fiorano ESB supports the JMS standard for communication and combines the management benefits of centralized hub-and-spoke architectures with the performance benefits of distributed architectures. The Fiorano ESB allows centralized control with fully distributed, parallel flows of data between participating Enterprise Services, with no single point of failure in the distributed infrastructure. Less scalable integration solutions support one or more centralized brokers, which tend to become data bottlenecks.

1.6.2 Data Routing Mechanisms

Participating services in a business process within the Fiorano ESB framework can communicate using several different methods for data routing, including:

- Traditional JMS routes, where data routing is a part of the business logic of the component(s)
- Content-based routing, where data is routed based on its content (typically XML-based)
- External routing, where participating services are completely oblivious to the routing of data between component instances. Such an approach allows easy reuse of components across business processes, because the process of routing data is completely disconnected from the process of producing the data; developers are thus shielded from data routing and can focus exclusively on business logic creation.

- Service execution, which can be central or end-point

1.6.3 Compute-intensive Services

The Fiorano ESB allows compute-intensive services such as data transformation logic, to be executed at the end-points of the network, allowing processing to be distributed for better scalability and performance. This is in contrast to traditionally centralized systems, where the central server tends to become a bottleneck to performance.

1.6.4 Orchestration and Management Tools

Basic middleware infrastructure typically provides Application Program Interfaces (APIs) to route data between instances of executing services. The Fiorano ESB implementation, however, provides substantially added value by allowing the composition of business solutions from pre-built, pre-tested enterprise services. Such composite application platforms, deployed on top of the Fiorano ESB infrastructure, simplify the componentization of existing Web services, database applications, legacy systems, and J2EE and .NET software assets. This also enhances reuse within enterprise business processes, driving down development and management costs.

1.7 Conclusion

By leveraging emerging standards for communication, connectivity, transformation, and security, the Fiorano ESB delivers a powerful, affordable, standards-based backbone throughout the enterprise and partner organizations. Fiorano ESB smoothes the operational path of the processes running a business and reduces the time, effort, and cost of integrating the different components supporting these process steps. A powerful benefit of this type of approach is that it allows in-house development teams to build new applications that are already “integration-enabled” and can easily be incorporated into the Fiorano ESB as required. Associated benefits include savings on investments of expensive skills, decreased time-to-market, and enhanced reusability of existing software assets.

1.8 Glossary

Term	Brief Description
J2EE	Sun's J2EE and Microsoft's .Net are the two dominant distributed computing architecture frameworks. J2EE provides portability of a single language (Java™) over multiple operating systems and hardware platforms.
LDAP	Lightweight Directory Access Protocol (LDAP) is a subset of X.500 designed to run directly over the TCP/IP stack. LDAP is, like X.500, both an information model and a protocol for querying and manipulating it. LDAPv3 is an update developed in the IETF (Internet Engineering Task Force), which address the limitations found during deployment of the previous version of LDAP.
. NET	Microsoft's .Net and Sun's J2EE are the two dominant distributed computing architecture frameworks. . NET supports a wide range of languages but is primarily tied to the Microsoft Windows operating system and Intel hardware.
Secure Sockets Layer (SSL)	An open, non-proprietary protocol for securing data communications across computer networks. SSL is sandwiched between the application protocol (such as HTTP, Telnet, FTP, and NNTP) and the connection protocol (such as TCP/IP, UDP). SSL provides server authentication, message integrity, data encryption, and optional client authentication for TCP/IP connections.
SOAP	Simple Object Access Protocol. SOAP provides HTTP/XML based remote procedure call capabilities for XML Web Services
UDDI	Universal Description Discovery and Integration UDDI provides a searchable registry of XML Web Services and their associated URLs and WSDL pages.
WSDL	Web Services Description Language. WSDL is an XML based Interface Description Language for describing XML Web Services and how to use them.
XML	XML has emerged as the standard format for web data, and is beginning to be used as a common data format at all levels of the architecture. Many specialized vocabularies of XML are being developed to support specific Government and Industry functions.
XSLT	Extensible Stylesheet Language Transform. Transforms XML document from one schema into another. Used for data interchange between systems using different XML schema, or mapping XML to different output devices.

Chapter 2: Architecture and Infrastructure Details

As discussed in the previous chapter, the Fiorano ESB is a new form of integration middleware based on new emerging standards for communication, enterprise communication, connectivity, transformation, portability, and security. The key benefits offered by an ESB include, faster implementation and deployment of new projects, a flexible platform for future expansion offering a high degree of reuse, seamless interoperability, and an improved return on investment.

The Fiorano ESB offers an easy route to enhancing, streamlining and automating business operations, together with end-to-end connectivity both internally and with third parties and consumers. The ESB allows users to draw on existing business logic and processes residing anywhere within the enterprise to rapidly assemble solutions for particular problems, leading to unmatched flexibility, increased productivity and improved responsiveness to changing business conditions. It provides a flexible base for future development, enhancing the reuse of Enterprise Service components across multiple projects.

Fiorano's unique ESB architecture combines linearly scaling performance and fault tolerance, allowing processes to be coordinated with transactional integrity across large numbers of applications running on heterogeneous platforms and protocols. In this chapter, we take a closer look at the Fiorano ESB architecture and the core features of the platform.

2.1 Fiorano ESB System Architecture

The Fiorano ESB implements a brokered, peer-to-peer (often referred to as 'super peer') system architecture, which combines the management benefits of typical centralized hub-and-spoke systems with the performance benefits of fully distributed peer-to-peer systems, while avoiding the particular disadvantages of both of these individual approaches. The following figure illustrates the Fiorano ESB brokered peer-to-peer architecture.

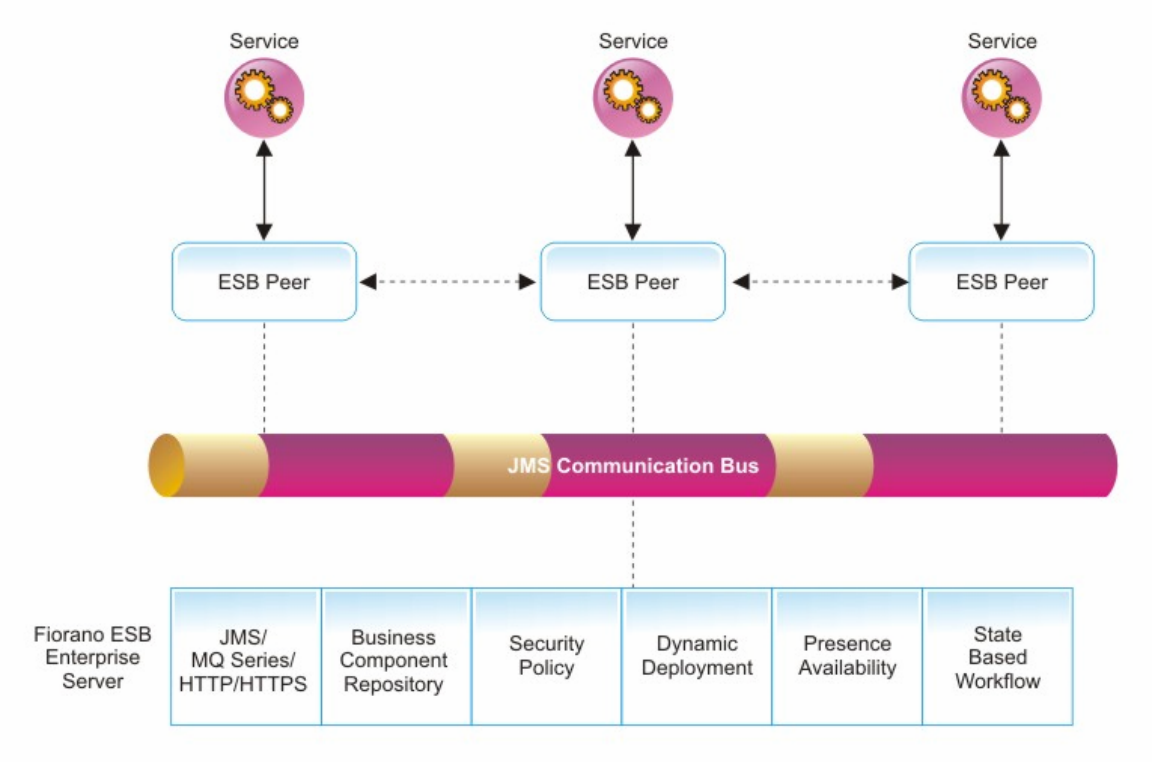

Figure 2.1: Fiorano ESB System Architecture

Particular benefits of the Fiorano ESB system architecture include:

- **Enterprise Class Backbone:** Underlying the ESB architecture is an enterprise-class, standards-based messaging backbone which provides secure and reliable communications between any number of applications and distributed processes across the enterprise. Using Fiorano's unique distributed peer-to-peer JMS implementation, the backbone allows distributed processes and composite applications to scale to meet the requirements of the most demanding global enterprise networks.
- **Efficiency:** ESB peers at the end-points of the network allow the components of a distributed application to exchange data concurrently, enabling all of the parallelism in a distributed business process to be exploited. For instance, an order management system in a manufacturing plant can check its inventory status even as the sales-force management system is updating the order database. Data transformations and other computations required by distributed business processes are performed concurrently at the end-points of the network.
- **Unbounded Scalability:** With dispersed computation and parallel data flow between nodes, ESB peers scale naturally and seamlessly with the addition of new peer nodes and Enterprise Services across the network.
- **Fault Tolerance:** Multiple peer servers at the end-points of the Fiorano ESB network ensure that there is no single point of failure. If one of the peers fails, Services connected to that peer fail-over to another peer in the system, allowing distributed processes to remain unaffected; in the case of a catastrophic failure at a peer node, the rest of the system continues operation, localizing losses to the failed node only.

- **Ease of Administration:** In a Fiorano ESB network, operations such as event-handling, security authentication and administration are performed by centralized servers. ESB peers at the end-points of the network are easily administered via tools connected to the centralized ESB Enterprise Server.

Fiorano's brokered peer-to-peer architectural approach thus combines the benefits of both peer-to-peer systems and hub-and-spoke systems in a single cohesive architecture for a scalable enterprise backbone.

2.1.1 Fiorano ESB Peer Server Architecture

ESB peer servers run at the end-points of a Fiorano ESB network, providing a platform for deployment and control of distorted enterprise processes. The Fiorano ESB Peer Server architecture is illustrated in the following figure.

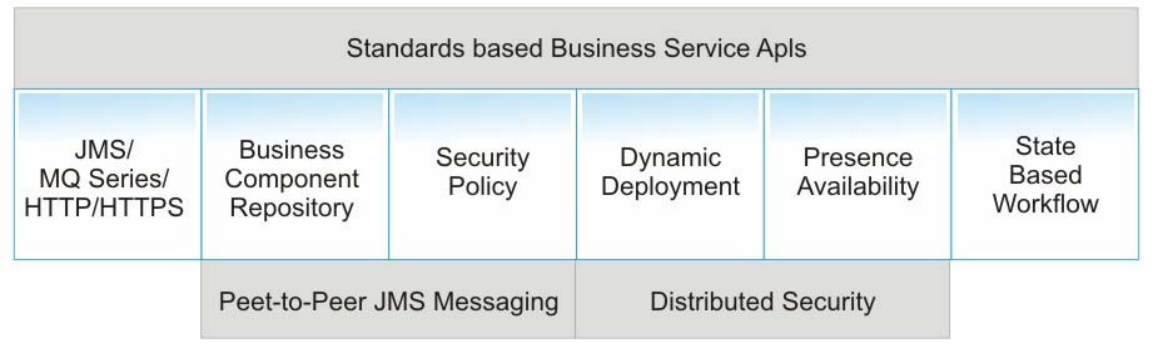

Figure 2.2: Fiorano Peer Server Architecture

ESB peers allow services to run at the end-points of a network, reducing reliance on expensive centralized servers and allowing efficient reuse of existing hardware resources. Each ESB Peer implements a set of enhanced ESB Services. While normal ESBs only implement support for XML, web-services, JCA and intelligent routing, Fiorano ESB Peers implement several value-added services including monitoring, scheduling, launching, remote deployment, tracing, logging, document tracking, presence and availability and several others as explained in the following sections.

2.1.1.1 JMS Messaging and Communications Services

JMS compliant messaging is implemented within each ESB Peer, allowing the creation of direct queues between network end-points, without the necessity of routing data through a central server. Such peer-to-peer messaging enables linear scalability of applications and allows enterprises to reuse the computing power at the end-points of their networks, in effect making the entire enterprise network a single Grid for enterprise distributed computing.

2.1.1.2 Transformation Services

Fiorano ESB Peers incorporate an in-built XSLT transformation engine, allowing XSLT transformations to be directly performed at the network end-points, improving the speed and scalability of distributed enterprise processes.

2.1.1.3 Intelligent Routing

In a Fiorano ESB process, routing of data does not have to be performed at a central control hub; instead, the routing information for a particular business process is passed by the central ESB Enterprise Server to distributed ESB Peers, and data routing decisions are thence automatically executed at the end-points of the network. This further enhances scalability and operational efficiency. Further, the routing framework within the Fiorano ESB does not require developers to manage any Topic/Queue or other destination hierarchies, further reducing the complexity and cost of deploying and managing enterprise-wide distributed processes.

2.1.1.4 Service Containers and Local Repositories

Fiorano ESB Peer at network end-points serve as a local repositories for Services that are deployed on that Peer. Each Peer can store any number of Enterprise Services in its local repository at the end-point of the network. Such local storage of Services allows applications to execute more efficiently, precluding the need for repeated deployments of services across the network at runtime.

2.1.1.5 Monitoring

Services within a distributed Fiorano ESB Process (or “Distributed Application”) always connect to a local peer. ESB Peers provide support to monitor locally running services; users can set up customized alerts in case a service goes down, start separate instances of a service on a remote node in case of such failure or publish monitoring events that can be captured on centralized tools (including industry-standard SNMP tools) for further processing.

2.1.1.6 Remote Launching

ESB Peers incorporate support for dynamic deployment of Services into the Peer at runtime. By dynamic deployment, we mean that all resources belonging to a particular Enterprise Service (such as, for instance, JAR files in the case of a Java-based Service, EXEs in case of a native service, DLLs in case of an Active X service, etc.), are copied from a central repository (typically residing on the central Fiorano ESB Enterprise Server) to the local ESB Peer. After all resources are copied, the Service can be launched optionally.

2.1.1.7 Tracing and Logging

A Service connected to an ESB peer typically logs data, especially in the case of error conditions. The Fiorano Service API allows developers to set customized trace and log levels at development time, and ESB Peers allow such levels to be dynamically modified at runtime. The ability to change trace levels in a running service (and thereby generate less, or more, logging information) greatly facilitates the process of debugging distributed processes and applications, reducing time to market and costs of development and maintenance.

2.1.1.8 Presence and Availability

ESB peers allow authorized applications/users to retrieve statistics about running and installed services to be retrieved from the Peer.

2.1.1.9 Security

Peer Servers implement a distributed security model. Security descriptors for Services and Applications (ESB Distributed Processes) are set at a central point (the Fiorano ESB Enterprise Server). ESB Peers implement the necessary support to allow/deny access to Services and Applications based on the security policies set by the Administrator. This mechanism allows fine-grained control over services and distributed processes executing across the network.

2.1.2 Fiorano ESB Enterprise Server Architecture

The Fiorano ESB enterprise server is a central point of control for composite applications and distributed business processes. The following figure illustrates the internal architecture of this server.

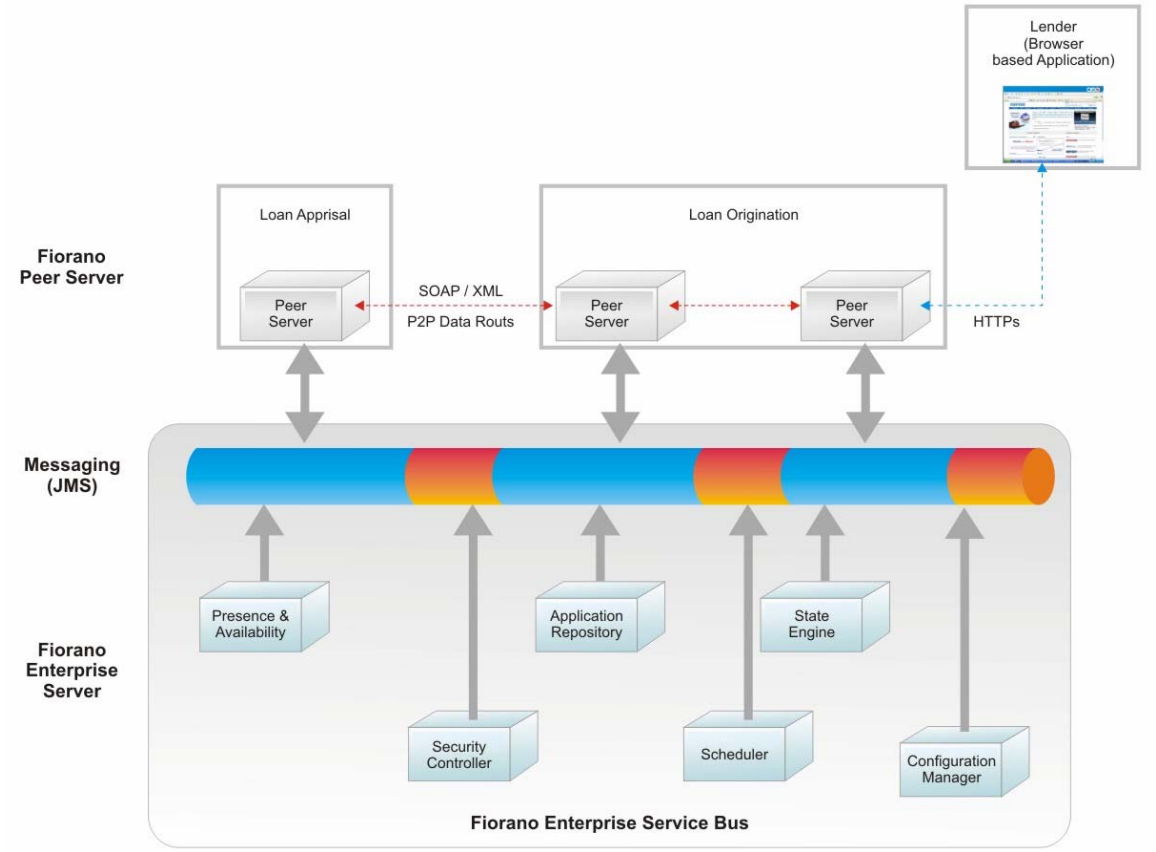

Figure 2.3: Fiorano Enterprise Server Architecture

2.2 An Overview Fiorano ESB Features and Benefits

As explained in chapter 1, the ESB forms the backbone of the entire corporate value chain, supporting all areas of activity and forming a mission critical element of corporate IT Infrastructure. So support this role, an ESB has to provide certain fundamental features, including

- Service-Oriented Architecture (SOA) based Application Composition
- XML, messaging, transformation, intelligent routing services
- Basic connectivity (Web Services, J2EE and .NET Connectors, JMS)
- Support for highly distributed deployments
- Manageability of distributed infrastructure and applications

In addition, key value-add characteristics of an ESB for mission-critical deployments include

- Robustness
- Scalability and Performance
- Security
- Breadth of connectivity

- Service-Oriented Development and Deployment tools

The remaining sections of this chapter discusses the details of how the Fiorano ESB implements each of these characteristics

2.3 Fundamental ESB Services

To qualify as an ESB, any infrastructure platform needs to implement certain fundamental services, which include:

- Communications-bus related services including transformation, XML support, intelligent routing and standards-based messaging
- Support for standards-based connectivity including Web-Services, J2EE and .NET at a minimum
- Tools to develop and deploy applications based on a Services Oriented Architecture (SOA)
- Administration and Management Services at both the infrastructure and application levels
- Adherence to industry standards for enterprise integration and business processing

The Fiorano ESB implements all of the above functions, as discussed in the following sections.

2.3.1 Service-oriented Architecture (SOA)

The model of distributed processing supported by the Fiorano ESB is based entirely on the notion of Enterprise Services (or simply “Services”). A Service is an application with one or more inputs and one or more outputs. For instance, a Web-Service is a Service identified by a URL, whose published interfaces are defined and described via XML. Services are typically coarse-grained and run in their own process-space. A Service may be developed in any programming language, and the Fiorano ESB supports development in Java, C, C++, C#, Visual Basic and even scripting languages such as Perl, JavaScript and Python.

Fiorano ESB Services can be deployed dynamically to multiple network nodes, much like software ‘agents’, and directly drive reuse since each Service has defined input and output ports with in-built schema validation, as further explained in section “XML Services on page 23” . Fiorano ESB Processes (also referred to as Composite Applications) are set up by interconnecting distributed Services with Routes. Routes are reliable data-flow channels which facilitate information exchange between Services for the performance of a particular business task. ESB Processes can be nested and reused within other ESB Processes.

The application architecture supported by the Fiorano ESB is thus wholly Service Oriented. The Service is a first-class object within the Fiorano ESB, and can be independently deployed, managed, secured and deleted from the system. Composite applications can be rapidly synthesized from existing Services (including Web-Services) allowing instantaneous changes to solutions to respond to changing business requirements’. Fiorano’s SOA approach improves systems development and maintenance productivity and drastically improves ROI by enabling the reuse of existing enterprise software assets.

2.3.2 Basic Communication-Bus Services

The Fiorano ESB includes a comprehensive, enterprise-class messaging backbone, with support for standards-based transformation, XML services and intelligent routing for the deployment of complex distributed business processes.

2.3.2.1 JMS Communications

The Fiorano ESB uses a standards-based, JMS-compliant enterprise-class messaging backbone, FioranoMQ, for all underlying communications between application and infrastructure components. With support both publish/subscribe as well as queuing, JMS has become by far the dominant API for distributed asynchronous communications within most enterprises. Unlike most other ESB implementations, the Fiorano ESB allows enterprises to plug-in any JMS-provider to serve as the underlying messaging backbone if required.

In addition, as explained in section “Support for Highly Distributed Deployments on page 24” the Fiorano ESB offers users the option of using a distributed peer to peer implementation of the JMS standard, dramatically increasing the performance, throughput and reliability of solutions.

2.3.2.2 Transformation Services

Transformation is a fundamental requirement for an ESB, since enterprise business processes typically involve data exchanges between distributed applications, and each application has its own expectation of data formats. The Fiorano ESB implements an native XSLT engine for standards-based XML transformation. GUI tools allow users to set up transformations without any XSLT programming, as illustrated in the following figure. Users can extend the XSLT mappings with custom code in Java and Javascript.

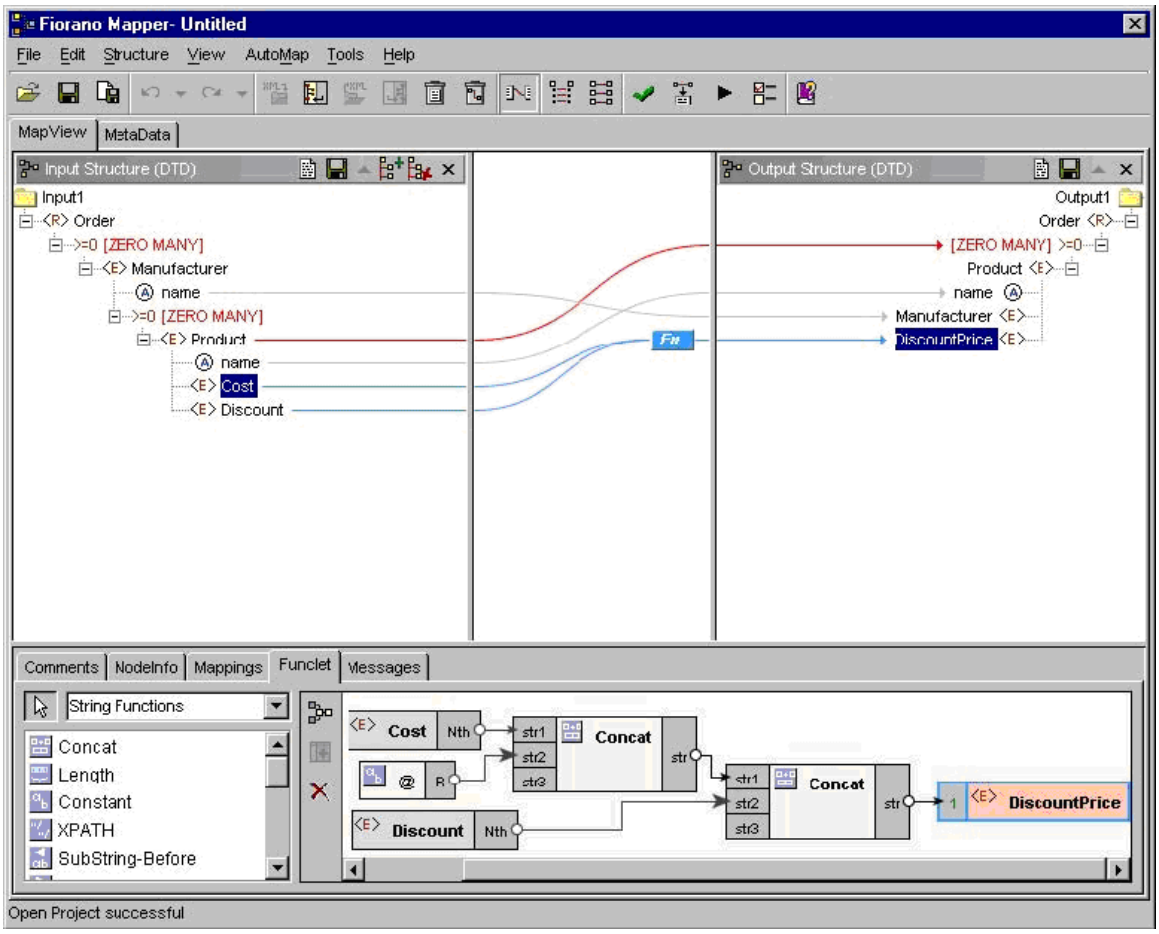

Figure 2.4: Fiorano XSLT Transformation Tool

The XSLT engine is also fully integrated into Fiorano's database and EDI adapters, allowing direct transformation of SQL statements based on JDBC and ODBC calls into XML.

2.3.2.3 XML Services

The Fiorano ESB incorporates native support for XML messages. XML messages of any length can be passed between distributed services. Further, XML descriptors (in the form of a DTD or XSD) can be associated as a property with each input and output port of an Enterprise Service, as illustrated in the following
figure.

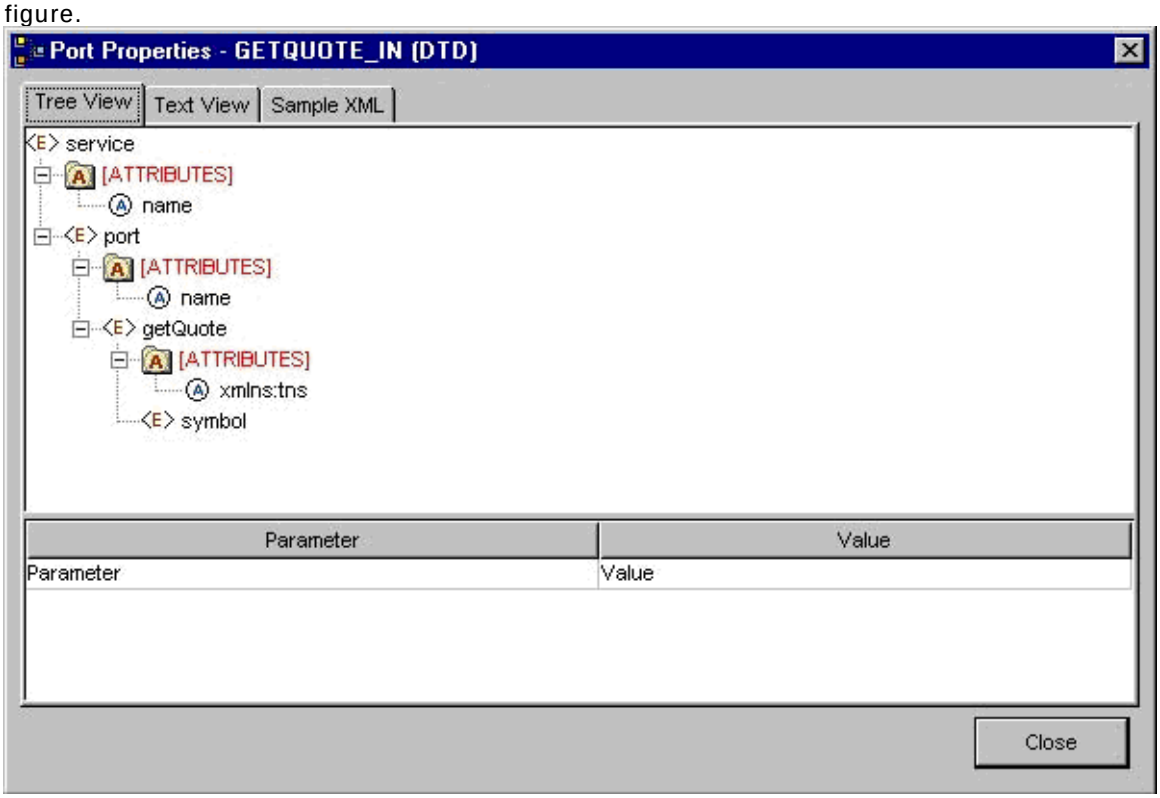

Figure 2.5: XML Assertions as port properties of Enterprise Services

Such XML assertions control the data formats entering (or leaving) the Service, allowing mismatches in data formats between distributed services to be easily detected. Relevant XSLT transformations can then be applied to ensure that all data flowing into a distributed Service is in the proper form.

2.3.2.4 Intelligent Routing

The Fiorano ESB allows the routing of data between distributed Services to be controlled externally, allowing users to set up business processes with ease. Importantly, the data routing is not embedded into the code that implements any of the participating Services. As such, the data flows between distributed Services can be changed by external tools, allowing easy, on-the-fly modifications to running business processes in response to external business requirements. Distributed ESB Processes are overlaid as a set of routes between participating Services, each of which could be running anywhere on the network.

The Fiorano ESB Intelligent Routing mechanism provides a key advantage over other ESBs in that it does not require the user or developer to keep track of the Topic, Queue or other “destination” hierarchy in the routing process. Data can be made to flow directly between software Services running on different machines: all queues, topics and other associated ‘middleware’ concepts are completely hidden from the programmer, allowing developers to focus on the business logic rather than on underlying middle-ware concepts.

Content-based routing, where the destination of a data route is determined by the content of an XML message, is also supported via special Content Based Routing (CBR) Services. The CBR Service intelligently processes the XML content of the information being passed from one step to another in a business process and chooses the appropriate next business step based on that information. This form of intelligent routing allows CBR Services to handle business rules effectively. Multiple CBR Services may be run at different end-points on the network, allowing business rules to be distributed for more efficient execution, while still providing a single point of control via centralized tools.

2.3.3 Support for Highly Distributed Deployments

The Fiorano ESB implements distributed process model that allows a single distributed process (also referred to as a Composite Application) to be composed of a set of Service instances, each of which run on a separate node on the network. As explained in “Fiorano ESB System Architecture on page 14” , Fiorano ESB Peers installed at the end-points of the network allow Services to be easily distributed. At the same time, control and administration are centralized within the Fiorano ESB Enterprise Server.

Location and Technology Transparency: Services plugged into the Fiorano ESB use the JMS API which provides natural location transparency: a Service an be run on any machine across the distributed network. The transport-layer within the Fiorano ESB can be any JMS Server, creating technology transparency at the transport-level; Further, Services themselves may be developed in a wide variety of programming languages, while the core infrastructure is Java-based for easy portability, creating the industry’s first multi-language, multi-platform ESB.

Distributed, Parallel Data Flow: Information and data flowing between distributed services does not have to pass through a central hub because each Fiorano ESB Peer incorporates JMS-compliant messaging server, allowing direct peer-to-peer connections to be set up on-the-fly between any set of peers across the network. This on-demand creation of peer-to-peer data-flow connections is unique to the Fiorano ESB and allows linear scalability and performance as new peers are added to the system. Furthermore, since peers can be hosted on existing (already very powerful) hardware at the end-point of the network, enterprises do not have to purchase expensive centralized hardware each time there is an increase performance requirement. The following figure illustrates distributed, parallel data-flow between Fiorano ESB peers.

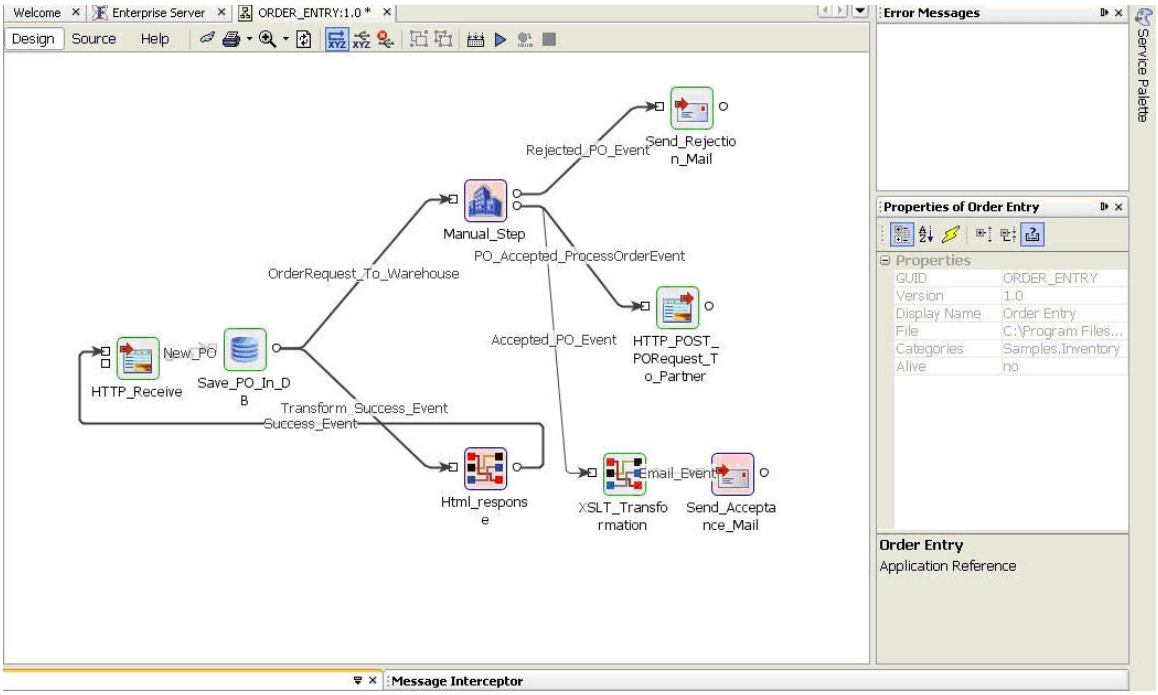

Figure 2.6: Peer-to-Peer data-flow across Fiorano ESB Peers

Centralized Control: While data-flow and routing is fully distributed, all Services and infrastructure modules within the Fiorano ESB network can be controlled from a single point - the Fiorano ESB Enterprise Server. The Enterprise Server supports administration tools that allow control information from across the network to be conveniently viewed and analysed. Manageability is enhanced via support for SNMP-based tools such as IBM Tivoli, HP-Openview and others, allowing a view of the entire ESB domain from one or many global locations.

Distributed, Dynamic Deployment: Services used in distributed processes are typically hosted on the file-system by the central Fiorano ESB Enterprise Server. At runtime, such Services can be dynamically deployed to any network end-point, the only requirement being that a Fiorano ESB Peer Server be located at the desired end-point to receive the Service. Dynamic deployment involves transferring all resources corresponding to the Service, including for instance JAR files and associated dependency files for Java Services, .EXEs for C/C++ Services, etc., from the central repository to the network end-point. All dependencies are stored by the Fiorano ESB Peer at the network end-point and available for participation with distributed business processes.

The dynamic deployment mechanism described herein is unique to the Fiorano ESB. Dynamic deployment is supported not just for Java but for Services developed in a va-riety of programming languages including C, C++, C#, Visual Basic and others. In addition to dynamic deployment, a Service can be remotely started, stopped, upgraded to a new version and/or uninstalled from the remote end-point. It is also possible to replace an existing Service with another one at runtime, allowing dynamic run-time changes to individual components of distributed business processes without affecting the rest of the process.

2.3.4 Basic Connectivity Services

The Fiorano ESB provides out-of-the-box connectivity to a wide-variety of enterprise systems via pre-built Services that ship with the platform. Bundled connectors include:

Web Services: The Fiorano Web-Services component automatically parses WSDL files, allowing the interfaces of any Web-Service to be exposed as input and output ports within an ESB-hosted Service. Any web-service can thus be instantly orchestrated within a distributed ESB process via point-and-click tools. The Fiorano Web-Services connectors work with a wide variety of hosting platforms from multiple vendors, shielding users from implementation differences and enhancing Web-Services interoperability.

Databases: The Fiorano Database adapter component allows out-of-the box connectivity to a wide variety of databases, allowing relational SQL data to be automatically converted to XML, thus providing an 'out of the box' Database-to-XML mapping function. Individual database tables may be monitored, and insert/update/delete triggers can be automatically or manually installed.

J2EE Connectors: The Fiorano EJB connector, preconfigured for several widely-used application servers, allows connectivity to any J2EE platform. Support for JCA allows a wide variety of JCA-compliant adapters to be used seamlessly within the Fiorano ESB environment.

JMS, MQSeries, MSMQ: Connectivity to any existing JMS-compliant server within the enterprise is provided via a JMS Service. MQSeries and MSMQ middleware connectivity is also via Service bridges.

2.3.5 Manageability

The highly-distributed architecture and asynchronous communications structure of the ESB make management of the overall infrastructure a particularly important task. There are three key aspects to managing a Fiorano ESB network:

- Administration and Deployment
- Monitoring
- System actions

2.3.5.1 Administration

Administration of the Fiorano ESB network is performed by tools connected to the centralized Fiorano ESB Enterprise Server. The central server serves as a repository for all Services used within distributed processes and applications, for all security information and other control information including control over all distributed processes and composite applications running across the network. The central server also has a view of all peers running across the network and can access all pertinent information about the peers, such as the Services currently hosted by any peer, the applications or portions thereof running on any peer, etc. Distributed processes can be composed, deployed, launched, terminated and extended via tools connected to the central server, making it a true single-point of control for the network.

2.3.5.2 Dynamic Deployment

Services within the Fiorano ESB network can be configured, upgraded and automatically deployed to any node in the network using integrated tools from the central Fiorano ESB Enterprise Server. Services participating in a distributed process can be replaced by different services dynamically at runtime without stopping the distributed process; the Fiorano ESB provides tools to ensure that the data formats accepted by the new Service match those accepted by the older one, ensuring that the integrity of the system is not compromised

2.3.5.3 Monitoring

Monitoring of both the infrastructure and of running applications, to ensure that corrected levels of service are being provided, is a critical task for an ESB. The Fiorano ESB implementation addresses monitoring issues in many ways:

Problem determination: The Fiorano ESB incorporates real-time alerts for all operations relating to distributed services. For instance, events are published to the bus when a remote Service is started, stopped, or suspended. A comprehensive monitoring and event API allows Services to publish customized alerts which are captured by the administrative console. Further, alert-levels can be adjusted within remotely running Services at runtime, allowing fine-grained control over the monitoring process.

Problem prediction: Alerts for common system problems such as disk-full, thread and process limits, etc. are built into the ESB infrastructure and can be controlled by developers via APIs. Additional problem predictions provided through runtime alert-level changes within distributed services, error ports within Services that allow errors to flow to special error-handling Services, and dynamically insertable distributed data-debugging capabilities that allow data flowing on queues and topics between distributed Services on a network to be examined 'in-flight' at runtime.

Support for Enterprise Management Frameworks: The Fiorano ESB supports the capture comprehensive system statistics relating to the ESB infrastructure itself as well as the hardware and software (operating systems, JVMs, and so on) on top of which it runs. Statistics can be captured both within the firewall as well as across company boundaries, provided that at least one ESB peer runs across the firewall. Notifications hooks are provided for SNMP-based enterprise-management frameworks such as HP Open-View and IBM Tivoli, allowing all ESB-related system information to be viewed within these frameworks.

2.3.5.4 System Actions

Once a problem or potential problem is identified, the ESB needs to provide a single point of control to resolve the problem or prevent its continued manifestation as an emerging problem. In the Fiorano ESB architecture, this point is the centralized Enterprise Server, which provides an overview of all distributed resources including Services and ESB Peers. Tools connected to this central server allow fine-grained control over running Services and distributed processes as follows:

- remotely running Services and entire distributed processes can be started and stopped as required

- data routes between distributed Services can be modified and changed at runtime via GUI tools without additional programming
- dynamic service-level tracing allows fine-grained, runtime adjustment of logging information
- message-editing and modification is provided via distributed debugging tools, with support for dynamic break-points

These tools and facilities dramatically ease the manageability of the Fiorano ESB, allowing problems to be prevented in most cases and allowing administrators to take immediate action when a problem is detected.

2.4 Fiorano ESB - Key Value-add Features

Besides the fundamental characteristics discussed in the prior sections, key value-add characteristics of ESBs include

- Robustness
- Scalability and Performance
- Security
- Breadth of Connectivity, and
- Service-Oriented Development/Deployment Toolsets

In the following sections, we examine how the Fiorano ESB addresses all of these points.

2.4.1 Robustness

Since an ESB represents a foundation platform for the deployment of distributed enterprise applications, it must provide the highest levels of robustness to avoid business risk to users. Robustness is typically addressed along two dimensions:

- Fault avoidance, ensuring problems do not happen
- Fault tolerance, ensuring that if and when problems do happen, they have little or no impact on service, or at the very worst are resolvable in a non-intrusive manner

While it is preferable to avoid problems altogether, in reality problems do happen. The Fiorano ESB allows problems to be handled quickly and effectively, as described in the sections that follow.

2.4.2 Fault Avoidance

The avoidance of problems is governed by three factors: the adoption of industry standards, ease of use of software tools supported by the ESB, and scalability.

Adoption of Standards: As explained in chapter 1, the ESB concept is based on a wide range of standards that have emerged over the past years, including XML, JMS, J2EE, Web-Services (including SOAP, UDDI and WSDL) and others, that cover a wide range of technical interface specifications. The Fiorano ESB implements all these standards, making it a lot more reliable and predictable compared to incumbent solutions since these standards have already been validated in the marketplace, proving that they work. Further, the implementation of the foregoing standards ensures that no specialist skills are required to implement solutions on the Fiorano ESB, reducing the likelihood of problems being introduced through a lack of technical understanding of the product.

Ease of Use: The Fiorano ESB incorporates easy-to-use, intuitive GUI tools for composing, deploying, monitoring and administering distributed Services and applications. The Fiorano ESB tools are unique in allowing business analysts and non-programmers to set up and deploy complex processes using pre-built, pre-tested services without any programming whatsoever. Enhanced fault avoidance is provided via sub-flows and error-flows embedded within the application-process flow. Since no specialist skills are required to actually compose business-flows, the Fiorano ESB avoids faults related to system usage that typically plague existing integration brokers and other ESB implementations.

Scalability: With its peer-to-peer architecture, the Fiorano ESB enables increasing loads to be seamlessly distributed across the network through the dynamic addition of peer servers. Since data flows between distributed processes are not routed through a central hub, the Fiorano ESB avoids load-related faults that plague most existing integration infrastructure solutions.

2.4.3 Fault Tolerance

When problems do occur, the Fiorano ESB ensures that they are resolved with minimum impact on running Services and disturbed processes. Fault tolerance within the Fiorano ESB is provided as follows:

Routing around failures: The Fiorano ESB implements dynamic message re-routing across distributed Services to avoid underlying network failures. In addition, store-and-forward semantics implemented within each peer ensures data integrity in case of unavailable paths between senders and receives.

Redundancy: Each Service running on a peer in the Fiorano ESB network can specify a backup peer for failover. In case the primary peer fails, the Service is dynamically deployed on the backup peer and all incoming data into the Service is automatically rerouted by the ESB to the new backup peer instance and fed to the newly launched instance of the Service. The Fiorano ESB is unique in implementing this ‘client-level’ failover.

In addition, the Fiorano ESB Enterprise Server supports High-Availability Clustered architecture with real-time mirroring of application state to ensure complete fault-tolerance. If the primary Enterprise Server fails, all connections are rerouted in real-time to the “hot backup” instance, with no impact on external applications.

Recovery: Fiorano ESB tools implement a compensating transaction model for recovery and long-running transactions. In addition, distributed transactions are also supported by the underlying JMS message-bus.

2.4.4 Performance

A high level of performance is essential to ensure that newly integrated and automated operations can be carried out effectively and efficiently, despite the inevitable spikes in demand for particular services. The Fiorano ESB implements Performance along many dimensions:

Asynchronous, Multi-Threaded Enterprise-Class Messaging Backbone: The Fiorano ESB uses FioranoMQ, a high-performance JMS backbone with support for asynchronous messaging and multi-threaded services. Centralized messaging servers may be clustered to deliver required performance levels. In addition, each Fiorano ESB Peer at a network end-point incorporates its own copy of the FioranoMQ JMS server, allowing distributed services across the network to exchange data concurrently via peer-to-peer JMS routes, thus ensuring that all of the inherent parallelism within a business process is exploited automatically.

The model of distributed processes implemented by the Fiorano ESB automatically ensures that while as many operations as possible run in parallel, the implicit serialization within certain business process flows is maintained. The data flows between applications are represented by graphs, and all independent trees within the application graph forest represent concurrent computations. Within a single tree, operations are serialized based on in-built dependencies.

Load Balancing: Load balancing supported at the Service level, allowing dynamic load-balancing to be added to running applications. For instance, it is possible to run three separate XSLT transformation Services on different ESB Peers and to route data using a round-robin or other load-balancing algorithm. Load-balancing of Services across different nodes can be set up directly by the end-user or administrator through intuitive visual tools and does not require programmer intervention.

Transparent Resource Addition: The Fiorano ESB allows additional ESB Peers to be added to the network incrementally, without affecting existing peers or running Services and distributed processes. In addition, Services can be dynamically deployed, or stopped, uninstalled and redeployed on any network end-point, allowing administrators to control and balance the overall load across appropriate enterprise hardware.

Performance Spikes and Large Data Handling: The underlying FioranoMQ JMS backbone which provides the transport for the Fiorano ESB implements sophisticated flow-control algorithms to handle peak-loads and to ensure that a single slow subscriber does not slow down other subscribers. The messaging backbone also allows very large messages (of unbounded size) to be transmitted and handled with ease via an internal message-streaming implementation.

2.4.5 Scalability

Scalability is critical in ensuring that an ESB can deal not just with current projects (likelyin themselves to be highly distributed, probably across company boundaries) but can also provide an extensible, adaptable platform for future growth. The Fiorano ESB addresses scalability issues as follows:

Transparent Resource Addition: The Fiorano ESB promotes a linear 'build as you grow' model, which allows an enterprise to add software resources in the form of Fiorano ESB Peers at network end-points to absorb additional load on the ESB. Peers can be added incrementally at runtime, without disrupting existing Services and distributed processes. Since ESB peers reuse existing enterprise hardware, resource addition typically does not involved additional hardware deployments, unless explicitly required.

Dynamic Change Support: Distributed processes and applications deployed on the Fiorano ESB can be extended and modified dynamically at runtime with the addition (or removal) of new Services and data routes. Existing Services within an application can be individually controlled via start/stop/upgrade/modify semantics, allowing incremental changes to distributed processes.

2.4.6 Security

With ESB integration projects spanning multiple departments and quite possibly companies, security is of major importance. The Fiorano ESB supports a number of security technologies for authentication and authorization as discussed in the following sections.

Access Control for Services

The Fiorano ESB implements ACL-Based, J2EE and LDAP compliant security for User and Service authentication. For instance, ACL's can be set to ensure that a particular Service or Component cannot receive data from (or send data to) another named Service. It is also possible to label Services, Applications and ESB peers with pre-determined string-labels and then control the deployment of labeled Services on labeled ESB Peers across the network. This allows the IT department, for instance, to set rules to disallow all (or a certain named set of) Services labeled "development" from executing on all machined (i.e. Peers) labeled "Deployment".

Security descriptors can be stored and retrieved from a standard LDAP repository, as well as in a Windows or Unix Realm.

Non-repudiation

Non-repudiation refers to the ability to be able to prove that a request really did come from an authenticated and authorized user or Service. The Fiorano ESB implements non-repudiation on data-routes between Services participating in a distributed process. Such non-repudiation is enforced via document tracking and audit-trails that are generated when a distributed process executes. The audit trails are automatically captured and stored in a relational database, forming a database of tracking information controllable at service-instance level.

Information Security

Information flowing between Services in a distributed process needs to be secured. The Fiorano ESB provides pluggable RSA and DSA encryption on all data-flow routes, allowing developers and users to set their own encryption standards via encryption and decryption Services. In addition, SSL-based level security is provided at the JMS transport level.

Tools Usage

The Fiorano ESB implements a comprehensive role-based security model to control access to and usage of all tools on a per-user basis. Administrators can group users into any number of groups, with user-rights assignment (much like Windows 2000 tools security architecture). The role-based security model ensures that only personnel with the required level of authorization can change system or application/service definitions or use any operational tools other than purely passive ones.

2.4.7 Breadth of Connectivity

In addition to the connectivity to existing enterprise systems such as JMS, WebServices and JCA, the Fiorano ESB provides additional connectivity to enterprise systems as follows:

- extensive third-party support for mainframe and legacy adapters, including CICS, IMS and Tuxedo
- native pre-built SAP adapters
- pre-built, native bi-directional bridges to existing enterprise messaging platforms including JMS, MQSeries, Tibrv and MSMQ
- native EJB adapter allows asynchronous invocation of WebServices and EJBs hosted on any J2EE compliant AppServer.

The Fiorano ESB also interoperates with other standards such as COM, CORBA, and .NET. While Fiorano ESB tools are shipped with Native COM and CORBA components for service development, they supports native .NET integration through C#, Visual Basic and ActiveX support. Native support is also provided for services developed in Java, C, C++, Perl and other scripting languages.

Finally, many ESB implementations are likely to extend outwards through some sort of Internet-based communications capability. This could be out to a portal or perhaps to some other Internet-based form of service. The Fiorano ESB supports HTTP Post and Get Services as well as SMTP, POP3 and FTP services, allowing direct connectivity to the more common Internet services.

2.7.8 Tools

The Fiorano ESB enables the achievement of integration aims quickly, easily, cheaply and with less risk. It offers tools and functionality in the areas of development and deployment to help achieve these aims, as discussed in the sections below.

2.7.8.1 Configuration

Fiorano ESB provides sophisticated, easy to use GUI tools allowing developers to easily create process definitions across distributed services with a drag-drop-connect metaphor.

Processes once configured and even deployed can be easily modified and changed. The modified configurations are dynamically and incrementally deployed across the network at runtime, with only the modified portions of a process getting affected by any changes. As with process definitions, process changes and dynamic redeployments are affected without any additional programming.

2.7.8.2 Connectivity - Wrappering APIs

Fiorano Services API provides wrappers that allow any legacy application - such as, for instance, a COBOL application running on a mainframe - to be exposed as an ESB Service. Wrappering APIs are provided in a variety of languages, including Java, C, C++, C#, Visual Basic and Active X. It is also possible to write wrappers in scripting languages such as Perl, JavaScript and Python to name a few. With the wrappering APIs, it becomes possible to expose literally any application as an ESB Service, since the only requirement for the creation of a Service is to wrap the inputs and outputs which is the precise task performed by these APIs.

2.7.8.3 Incremental Deployment

The Fiorano ESB wrapping API and component model allows existing legacy applications to be incorporated as Services in a completely non-intrusive manner. Location transparency is built into the component model, since data routing information is completely detached from the Service Code. As such, Services can easily be executed anywhere on the network, allowing incremental deployment of Services and the distributed applications and processes within which they participate.

The Service-based approach of the Fiorano ESB simplifies componentization of existing Web Services, Database, Legacy, and J2EE and .NET software assets, enhancing reuse within enterprise business processes. Available Fiorano ESB Tools manage multiple versions of components, allowing version upgrades to be driven from a centralized location via GUI tools.

2.7.8.4 Life-cycle Management

The Fiorano ESB allows administrators and deployment engineers to label components into various states, for instance: Development, QA, Staging, and Production. Such component-state labeling, 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. For instance, with this mechanism it is possible to prevent any Component labeled “Development” to run on any ESB Peer labeled “Production”. The labels can be applied to named components only, or as part of named applications, allowing a wide variety of deployment options.

The Fiorano ESB configuration management tools also allow Services to be ‘moved’ between labels, with appropriate actions executed as part of the transition. For instance, a Service moving from “Development” to “QA” might have its default deployment machine descriptor changed to that of a QA machine rather than a Development machine. In the same way, entire applications and distributed processes can be moved through the lifecycle - Development, QA, Staging and Production. Note that there is no limit to the number of labels, so individual teams can set their own confirmation management options.