



Enterprise Application Integration

A Primer in Integration Technologies

By

John LaFata, Director
Scott Hofmann, Technology Manager

Topics covered in this paper include:

- Defining EAI
- The Value of Integration
- EAI Architecture Components
- EAI Technologies and Techniques
- The Evolving EAI Landscape



Overview

The evolution of integration technologies is undergoing a major transformation with the introduction of new technology solutions as well as new methods, standards, and practices. The emergence of the Application Server Platform Suite and Web Services are poised to take the solutions focused on EAI one step closer to the dream of a complete enterprise integration model. By employing these new platforms and approaches while adhering to a set of guiding principles critical to any integration effort, the modern enterprise has its best tool set yet developed to reach its business and application integration goals.

Enterprise Application Integration (EAI), aka “middleware,” provides the infrastructure to connect information sources, acting as a go-between for applications and their business processes. In implementing EAI solutions, organizations have been able to realize various benefits, including:

- Reduced development and maintenance costs
- Enhanced performance and reliability
- Implementation of a centralized information bus
- Extension of the legacy system lifecycle
- Reduced time to market

2. The Value of Integration

Early applications running on monolithic systems served individual target departments well, but information exchange between systems was limited. The evolution of client-server technology and improved relational data management technologies expanded the tools available for integration. The introduction of these new technologies to tie stand-alone applications together added complexity but also heralded the beginning of true Enterprise Applications, including Enterprise Resource Planning (ERP). ERP was built as an integrated suite of functions that allowed business processes to be modeled across departmental boundaries.

In the late 1990's, the commercial expansion of the Internet gave enterprises a common communication channel that created additional collaborative demands on systems.

This period saw the emergence of other enterprise applications such as CRM and SCM solutions, which needed to be tightly integrated across the enterprise. At first, enterprise software vendors did not include connectors to integrate their products with other packages, creating a market for independent software developers to introduce EAI middleware to facilitate this integration.

The driving force of application integration is the ability for companies to transform their business through the automation and integration of businesses processes. Addition-ally, companies are expanding to a value chain approach, including customers and suppliers. EAI promises to help organizations better perform these functions.

Several business drivers have driven integration efforts, including the need to:

- Improve the visibility of information across functional and corporate lines by linking together disparate systems with-out having to develop custom interfaces
- Reduce data entry time, costs, and errors
- Improve application performance by creating event-driven business processes with lower latency times
- Introduce common "integrated" services used across enterprise systems, such as security and transaction processing, to reduce operating and support costs
- Decrease application migration costs during the transition to a new system or application
- Provide a common infrastructure for the merger of two entities

While business process integration throughout the value chain can produce huge benefits for companies, the path to get to these benefits can be difficult. There are three major problems that companies can run into while utilizing an EAI system:

1. There is a dichotomy of modern application architectures, with developer following Sun's Java 2 Enterprise Edition (J2EE) and Microsoft's .NET
2. There is a trend within IT to purchase "packaged" applications to satisfy niche business requirements
3. The development of a wrapper for legacy applications, which may have no integration interface at all, is required to facilitate an integrated architecture (Roch 2)

Enterprise Application Integration soft-ware suites evolved to tackle the complex requirements of application integration, providing key areas of functionality, including:

- An integration broker (providing a set of services for message transformation and intelligent routing)
- Development tools for specifying transformation and routing rules and for building adapters into applications
- Off-the-shelf adapters for popular enterprise packaged applications (e.g., SAP R/3)
- Monitoring, administration and security facilities
- Message Oriented Middleware (MOM)
- Business process managers, e-commerce features and portal services.

When envisioning an overall EAI solution, an organization must investigate and focus on the overall business integration needs and goals as they relate to both internal and external constituents. In doing so, the organization will need to view integration from application-to-application (A2A) and business-to-business (B2B) perspectives. An A2A integration approach has a more narrow internal focus based on its goal of creating a single enterprise model that comprises all of the technology assets within the boundaries of the enterprise.

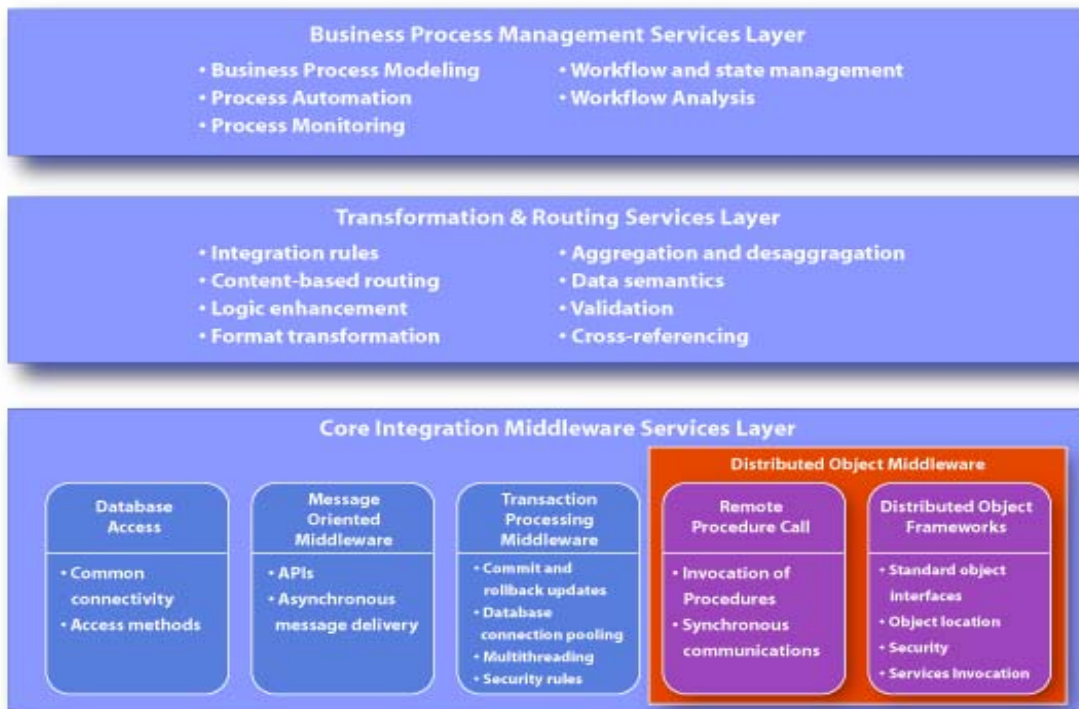


Figure 1: The EAI Functional Model

The A2A approach typically includes a common application architecture and security model, enabling a uniform approach to application development and integration. In addition, A2A integration incorporates centrally controlled access schemas, again creating a common operation and support model for all applications within the organization. These common elements, policies, and procedures can be developed and enforced efficiently, as all applications in questions are controlled by the organization itself.

With B2B integration, developing a single technology model is usually impossible due to the fact that multiple organizations, each with their own technology model, are involved in the overall architecture. In designing B2B technology architectures, organizations must differentiate between public and private components. This requirement along with inter-organization security and tightly controlled information and application access schemas are imperative to developing a secure inter-enterprise

architecture. Additionally, parties included in any B2B endeavor should include within their internal architectures support for common protocols.

3. EAI Architecture Components

At its core, EAI solutions can be categorized into three basic layers that make up the majority of technologies common in today's integration solutions. Each layer represents specific forms of EAI while also acting as services for solutions based on the layers above. These layers of abstraction form the building blocks of the EAI technology stack enabling the creation of more powerful solutions as you go further up the EAI stack.

Making up the foundation of the EAI functional model is Core Middleware. This layer represents the plumbing that allows multiple methods of application-to-application communications. Core middleware techniques and methods can be incorporated directly into the applications that need to communicate with one another. Core middleware, as the name implies, makes up the foundation with which most EAI solutions are built and includes database access routines, message-oriented-middleware (MOM), transaction processing middleware (TP), remote procedure calls, and distributed objects.



Figure 2: The EAI Functional Model

The message transformation and routing layer builds upon the core middleware foundation with a wide array of components that can further exploit the technologies



mentioned above. This layer includes tools to manipulate data contained in messages between applications. As an application generates a message, components in the message and transformation routing layer receive, re-view, revise, and reroute the message based on a set of rules predefined within the environment. By providing these services, applications do not need to include message queuing, data type matching, and application routing functionality. Instead, application developers can use the same mechanisms across all applications through the use of solutions that fall within the message transformation and routing layer. In addition to message brokers that are the common solution component in the layer, application servers have also begun to provide message transformation and routing services through message broker service components.

Most recent to emerge and extend the EAI functional model is the Business Process Management (BPM) Services Layer. This layer contains tools and components that allow for the modeling of discrete business processes across multiple applications. By providing a layer of abstraction above applications themselves, BPM allows a modeler with little technical knowledge or expertise the ability to map out the appropriate movement of data across many applications, possibly at many different organizations, in order to complete business tasks. Although at its surface simple, the complexity resides in the set up and mapping of applications and procedures within the tools themselves. Appropriate communications channels and middleware components from the other two EAI layers must be in place before a BPM solution can be successfully implemented and used.

4. EAI Techniques and Technologies

4.1. Core Middleware Services Layer

Database Access

The most primitive and widely implemented manifestation of EAI is database integration. This type of integration provides access to multiple, disparate, geographically separated databases. Database access middleware may also provide a common API, allowing multiple applications to utilize the same data access methods.

Examples of database access middleware include Open Data-base Connectivity (ODBC) and Java Database Connectivity (JDBC).

Message Oriented Middleware

In its basic form, messaging allows two entities to communicate by sending and receiving messages. In general, Message Oriented Middleware (MOM) products pass information in a message from one application to one or more other applications. These solutions can be configured in a point-to-point or multipoint topology such as a publish-and-subscribe model and provide fault tolerance and delivery guarantee. The introduction of message queues allows the producer and consumer to work at different paces. For example, a purchase order generated by an inventory system can be placed on the queue that is served by a vendor's fulfillment system even if the fulfillment system is down for backup. When the fulfillment system comes back online, it can process the order and acknowledge its receipt with a message in another queue to the purchaser. An-other common component of MOM based solutions is an Application Program Interface (API) that abstracts the network implementation of the application from the developer. Modern MOM technologies and features include:

- Message multicast (supporting one to many delivery)
- Reliability and serialization of messages
- Subject based addresses to abstract the network implementation
- Real time data delivery via multicast
- Support for multiple communications models
- Transactional boundary support

Companies such as IBM and TIBCO have popular message oriented middleware solutions on the market, both using proprietary components to accomplish similar tasks. However, the Java Message Service (JMS) has been introduced as a specification within the J2EE framework. JMS is an API for messaging that addresses many features typically found in stand-alone message oriented middleware products.

Transaction Processing Middleware

Born out of the mainframe application model, Transaction Processing middleware (AKA "TP Monitor") handles resource management and transaction coordination in a distributed system. TP Monitors provide many services, including application startup and shut down capabilities, transaction identification and demarcation, security administration, database connection pooling, common interfaces to network resources, multithreading, and message-based communication.

Above all, transaction processing middleware provides commit and rollback functionality. When a particular transaction updates multiple databases in order to successfully complete, assurance must be made that all databases have been updated properly before the system reaches its end state. TP monitors have the ability to hold the original data until such time as all proper data updates are made. At this point, the TP monitor will commit the data and end the transaction. If however any piece of the transaction should fail, the TP monitor will roll back the data so that all affected systems are returned to the state they were in prior to the beginning of the transaction.

Distributed Object Middleware

Remote Procedure Call

Remote procedure calls (RPCs) provide the ability to invoke a function from within one application and have that function execute within another application on a remote machine. To the developer of the application, functions executed in this manner act as if they were being executed on the local machine in the current application. This type of approach is synchronous in nature and requires a tremendous amount of processing power and network resources. For this reason, RPCs simplicity is many times outweighed by its limitations and overhead costs.

Distributed Object Framework

Although they provide a means for inter-application communication, Distributed Objects are at their core a mechanism for application development. From a programming standpoint, distributed objects provide a means of method sharing across the enterprise, using standard interfaces and protocols to communicate between



objects that make up applications. Through the use of standard frameworks such as COM, CORBA, and J2EE, objects can be created using a standard that inherently enables the objects to exchange information and carry out application functions by invoking each other's methods.

4.2. Transformation & Routing Services Layer

Integration / Message Broker

Integration brokers, also known as message brokers, act like a combination of a postal service and a translation service within application communication channels. The efficiencies of Integration Brokers can be easily explained by imagining the cost benefit of the postal service compared to the cost in time and resources of you hand delivering each package and letter you ship.

With integration brokers, each participating application creates an interface to the broker once. In addition to delivering the message, the broker translates the message into the appropriate format for the recipient application. This allows developers to programmatically reformat and route information moving from one system to another and to minimize the impact of change on both the source and target systems, which in turn reduces the overall technical cost while maximizing flexibility.

The integration broker encapsulates the integration process logic associated with application integration. By doing so, the integration logic, including the sequencing of events and data flow, is moved out of the application where the business logic is defined, ensuring the integrity of the business transaction. The integration broker also coordinates the sequence of integration steps that consist of checkpoints and transaction boundaries that guide the integration of data. As business rules change, the process definition can be changed within the broker development environment. This reduces application maintenance and provides business flexibility. When combined with the workflow capabilities of a BPM engine, business processes can be tailored to specific business events or customer/supplier classes optimizing supply chain execution.

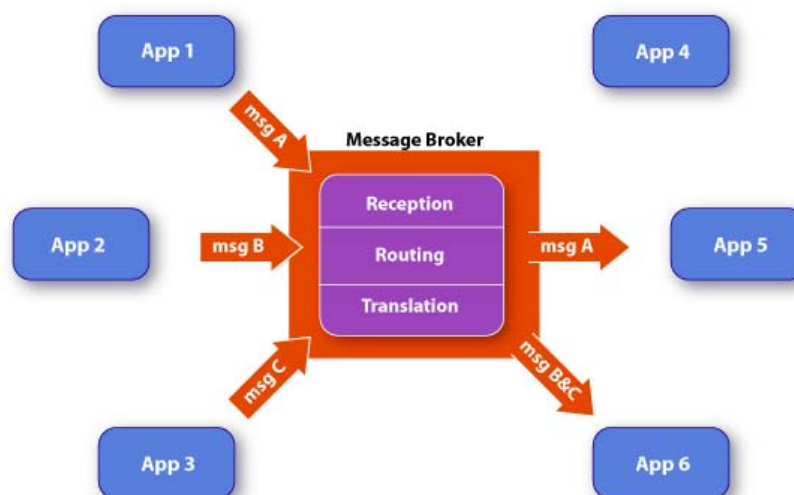


Figure 3: Illustrates where the broker fits between applications that are integrated using this approach.

What separates one broker's product from another are the flexibility they provide to interface with systems and the workflows that they support once the message is delivered to the broker. Imagine a Purchase Order that may originate in an ERP system. One particular vendor may provide an adapter to that ERP system's purchasing module and a graphical tool to manage the transformation and routing on the broker component. These features may help reduced the time it requires to interface with the broker system and may present a case to acquire this vendor's tools.

A few key features of integration brokers include:

- Transformation that captures schemas and then graphically maps input to output
- Intelligent message routing with check-point and transactional support
- Graphical design tools used to design transformation, process, and routing rules
- Distributed architectures
- Repository to store metadata regarding transformations, processes, and message routing to encourage process reuse
- Out-of-the-box processes including XML parsing, Encryption, and SMTP email

- Support for Object Request Brokers (ORB) and application component models such as:
 - Microsoft COM/COM+ and .NET, J2EE Enterprise Java-Beans (EJB) and Web Services
- Support for various MOM standards:
 - TIBCO's Rendezvous, IBM's WebSphere MQ, Microsoft's Message Queue (MSMQ) and J2EE Message Service (JMS)
- Support for legacy EDI data formats and XML data exchange schemas such as:
 - Rosetta NET, cXML, BizTalk, XEDI and ebXML

Below, Figure 4 depicts many of the associated technologies and standards that are used in today's integration broker centric EAI solutions.

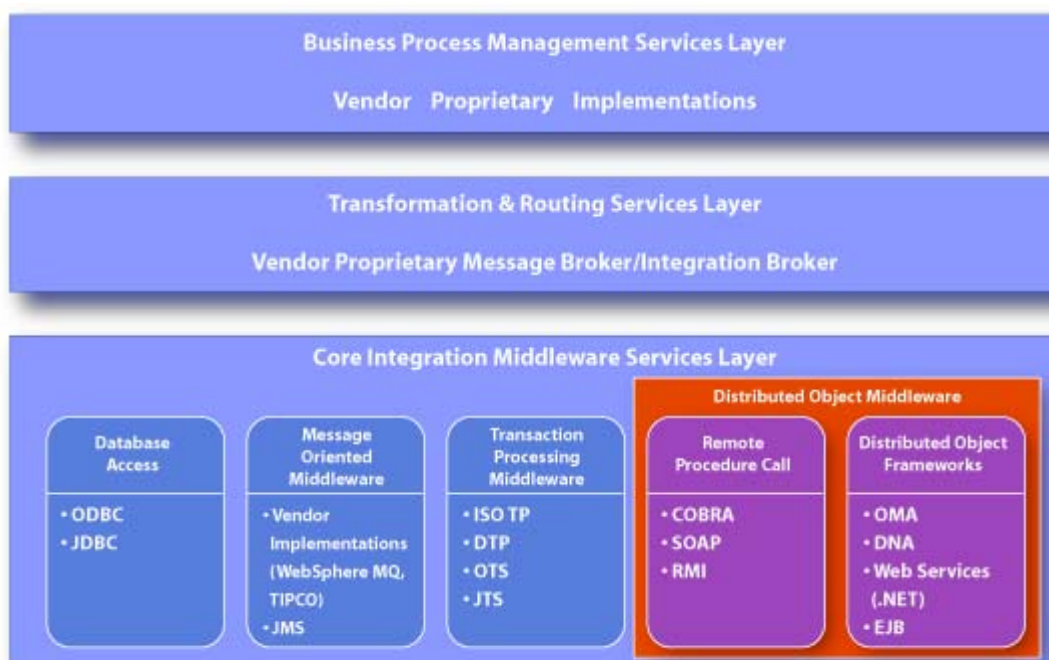


Figure 4: Message Broker EAI Model

Application Server

Repeated overhead development efforts have been plaguing application development for years. No matter the application, certain common components such as data connectivity, security, and synchronous integration methods needed to be custom built for each new application. This issue gave rise to Application Servers, which provided support for these common functions. Application server vendors provide the infrastructure for developers to manage information, represent business logic, and

interact with users, including other systems, while isolating the users from the fact that these systems may be composed of a variety of internal and disparate applications. Application Servers generally provide core object and session management functionality through the introduction of Object Containers.

A traditional application server may sit next to, or completely replace the integration broker in the message transformation and routing services layer, depending on the application integration requirements of the enterprise. Integration brokers are better suited for asynchronous messaging, while application servers can provide robust and efficient synchronous communications.

4.3. Business Process Management Layer

Business Process Management allows for the modeling of complex business processes and inter-application transactions without the designer having detailed knowledge of application source code and integration points. BPM provides a modeling tool to visually construct, analyze, and execute cross-functional business processes. Business processes are typically meant to be relatively long-lived processes that require both automated and human tasks. BPM executes automated tasks by defining process definitions within the integration broker component and human tasks via a defined workflow. Processes are value-driven, which allows BPM to analyze, define, and enforce process standardization. Processes are then decomposed into sub-processes, which make up individual business tasks. In all, BPM tools help empower the business analyst and modeler in their quest to create the most efficient and effective operational model for an organization.

5. The Evolving EAI Landscape

Until recently, many of the integration techniques discussed earlier had an associated technology that brought theory into reality. These point solutions, such as message brokers or application servers, provided desperately needed functionality to the IT organization. However, with many great technology concepts, the diversity of vendor solutions coupled with constantly changing standards caused a gap in the dream of true organization wide EAI. There are numerous vendors that supply application

servers, message brokers, and business process management tools. Still others have entered the pure play EAI market providing integration solutions that combine other integration technology concepts through proprietary implementations.

Although the EAI market is in a fragmented state, some emerging standards are beginning to drive towards a common platform capable of true enterprise-wide application integration. These standards, while revolutionizing the EAI and IT landscape, are also a by-product of their rise in popularity allowing one particular technology, the application server, to surface as the primary vehicle for EAI.

5.1. XML

Based upon the Standard Generalized Markup Language (SGML), the Extensible Markup Language (XML) is a close relative of HTML. Unlike HTML, which defines the presentation of data; XML defines the content itself, the mission critical data organizations need. By encapsulating both the data structure and the data itself, XML provides a common data exchange format allowing disparate applications to exchange information without each application knowing inherent logic and data types utilized by each participant application.

One critical factor in the use of XML-based data exchange is the adoption of standard vocabularies to be used for all XML messages. As with many standards-based approaches, organizations will have to make some decisions when choosing the appropriate vocabulary. In particular, competing technology providers will continue to create separate business-specific XML vocabularies that alone will not be inherently compatible. Organizations must choose one standard, or incur the costs of designing systems that can use multiple standards.

Furthermore, XML alone cannot solve all enterprise integration issues. XML is a standards-based language that deals solely with physical data and its structure. Transport mechanisms must be in place in order to deliver XML encapsulated data to its target applications. Knowing this, most EAI vendors support XML based information exchange. As standards continue to arise and become more universally accepted, XML-based data exchange will allow EAI solutions to be designed and developed more effectively.

5.2. Web Services

Web Services define another approach aimed at solving the same fundamental problems outlined in this paper. At its core, EAI is focused on allowing multiple disparate applications to work together, sharing data, processes, and other critical functionality across the enterprise. Web Services expand on this idea by doing away with middle-tier message brokers and instead exposing application functionality as services. These services are made available from other applications for synchronous invocation, similar to API calls. In order to achieve this, an organization must focus on creating a service-oriented architecture (SOA). An SOA is simple in concept, but complex in reality. Fundamentally, SOAs are focused on wrapping existing applications with a well-defined interface that transforms the application into a set of services accessible by other applications. In order for an SOA to meet this goal it must address a few key components. These components include:

- A common, universal interface language used to wrap the applications
- A ubiquitous and universal information transport mechanism
- Flexible, evolving interfaces
- Standard mechanisms to search for, locate, and execute services

Web Services address each of these issues comprehensively. By defining standards based on XML, web services are well suited to meet the overall goals of a SOA. These standards include:

- Simple Object Access Protocol (SOAP) - SOAP is a protocol for messaging and RPC-style communication between applications. It is based on XML and uses common Internet transport protocols like HTTP to carry its data.
- Web Services Description Language (WSDL) - WSDL is an XML-based description of how to connect to a particular web service. A WSDL description abstracts a particular service's various connection and messaging protocols into a high-level bundle and forms a key element of the UDDI directory's "green pages."
- Universal Description, Discovery, and Integration (UDDI) - UDDI represents a set of protocols and a public directory for the registration and real-time lookup of web services and other business processes.

Applications are developed and application descriptions are produced in WSDL. The WSDL t-schemas are published to UDDI directories. Clients search the UDDI directories to obtain the services they want to invoke. Communications channels via the Internet are bound and the services are invoked using SOAP. In this manner, the underlying design of applications and the programming languages used become inconsequential. Any service can be exposed using these common interfaces, fully utilizing organizations' technology assets.

5.3. The Application Server Platform

The application server has, since the late 90's, emerged as the primary development and runtime environment for modern applications. The emergence of the J2EE and .Net distributed component models has fueled the explosive growth of the application server market as organizations strive to find a common technology platform with which to build its application backbone. Now, with the proliferation of XML and Web Services, the application server market is once again poised to grow and prosper in the new age of EAI.

EAI methods and supporting technologies have sprung up as new needs and demands surface. Yet the application server has become the nucleus of many enterprises' IT infrastructures. Now, the application server market has realized their position as the gate-way for application development and deployment and responded by developing new capabilities that will enable robust EAI. Application server vendors are focusing their core server products into a foundation layer upon which EAI and BPM services can be added. This approach leverages the application server's function as a transaction management, database access, and business logic execution environment while expanding its role to encompass application-to-application integration within the enterprise and between trading partners, as well as business process modeling, integration, and automation capabilities.

Already, application server vendors have taken aggressive steps to incorporate EAI tools into their platforms. By utilizing the J2EE service component framework,

application servers can run packages that have classically been stand-alone EAI solutions as services within the overall enterprise architecture. BEA has introduced the eLink integration server and WebLogic Integration, which together provide application integration and business process management functionality. IBM has introduced this concept into its WebSphere application server platform with WebSphere Business Integrator, which adds EAI, Business-to-Business (B2B), and business process management functionality to its core application server solution. Additionally, by supporting the Java Connector Architecture, application servers provide a common model upon which connectors may be built that allow connectivity and integration into any JCA supporting application or system. Through the use of this connector technology, application server vendors, third-party software providers, and internal IT shops can follow one standard with which applications can be integrated, or connected, into the overall enterprise architecture.

The diagram below, Figure 5, illustrates the Application Server Platform and its use of J2EE, service components, and connectors to fulfill many of the overall goals of EAI.

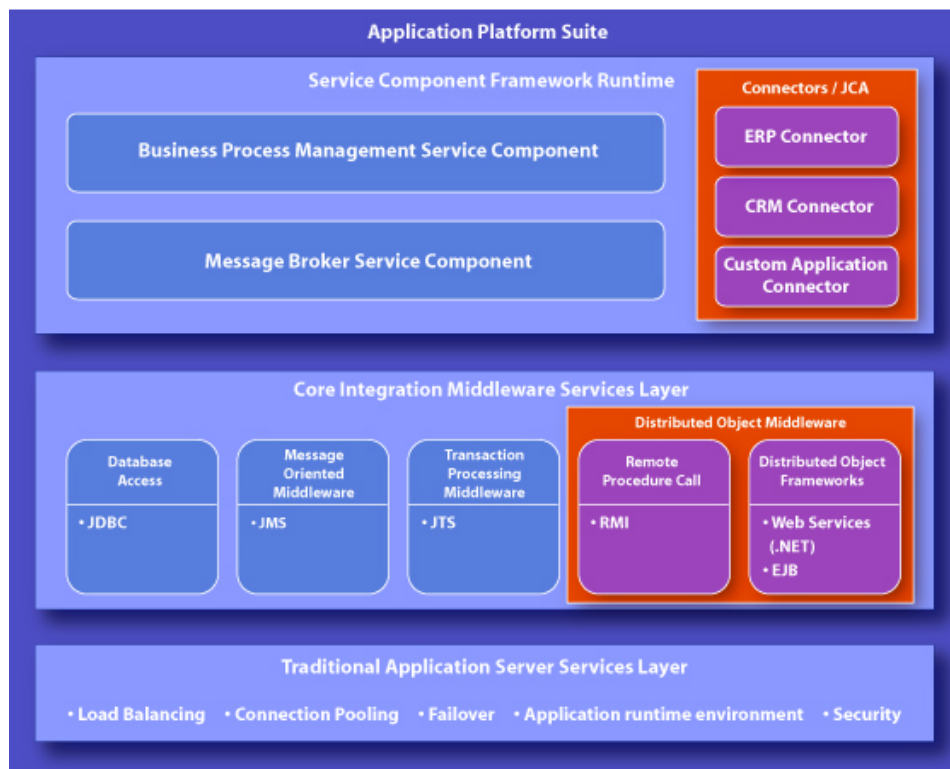


Figure 5: The Application Server Stack

Conclusion

Application server vendors are extending their products to adapt to ever-changing integration standards. The two key emerging standards that currently have an impact on application integration and will mold the space into the future include Web Services and the J2EE definitions for the Java Message Service (JMS) and the J2EE Connector Architecture (JCA). However, this emphasis on Web Services and J2EE should not downplay the importance of legacy standards and de facto EAI tools and techniques. It is critical that the application server platform support preexisting standards such as CORBA and EJB as well as common proprietary systems such as COM, MQ Series and CICS (Roch 12).

No matter the journey, one must plan their steps carefully in order to reach their goal. The same is true for implementing an Enterprise Application Integration solution: clear ownership of the overall program and identification of the business case must be established. Comprehensive business process and technology architecture reviews must be conducted. Consolidation and integration should be executed in steps. Management of an EAI solution's design, development, and deployment have equal, if not greater importance than the solution itself. Post deployment periods must be tightly controlled to ensure current and future viability of the integrated environment. Appropriate technologies must be brought in that support the business need, not necessarily the technology want. Of these technologies, Application Server Platforms stand ready to provide a common, standards-based foundation upon which highly integrated applications may interact. By leveraging existing business and technology assets and resources, and supplementing where appropriate with new business processes, technologies, and the resources necessary to get the job done, true enterprise integration has become more realistic and achievable.

References:

Linthicum, David S., B2B Application Integration, Boston, San Francisco, Addison Wesley, 2001

Technology Forecast: 2002 from Pricewater-house Coopers

Roch, Eric. Application Integration Business Case and Technology Trends, TIBCO, 2002



Phone: 484.654.1400, Fax: 484.654.1401, Email: information@liquidhub.com

About the Strategic Technology & Advancement Research (STAR) Team

The Strategic Technology & Advancement Research Team serves as the proving ground for our ideas.

STAR brings together the best and brightest of LiquidHub to encourage new thinking and develop through leadership around technology. Breakthrough ideas emerge and are exercised by the Team, to strengthen client engagements and educate industry leaders.