



BeePlex™

## Whitepaper

### Abstract

BeePlex implements all the tools necessary to run a mission critical Service Oriented Architecture (SOA) including security, centralized management, built in monitoring and fail-over capabilities in addition to extensive statistical reporting.

BeePlex EDA uses a "message bus" and XML messaging which allows you to implement rich work and process flows. Reliable store-and-forward messaging guarantees delivery of events and ensures that your system is robust. Because BeePlex provides you with an inherently decoupled architecture it gives you unparalleled flexibility and component reuse. Incremental development of your integration solution is greatly facilitated. BeePlex is quick and easy to deploy and allows for the versioning of components.

Add ZigBee wireless technology to this and you have an enterprise ready wireless solution that is truly scaleable with unparalleled adaptability to ever changing and challenging requirements.

### BeePlex and Enterprise Application Integration (EAI) Overview

Up until about mid 1998, EAI was generally perceived as being point-to-point integration. After that time a proliferation of integration products have emerged in response to the growing industry preference for brokered integration architectures. GartnerGroup says that they are currently actively tracking ten messaging middleware products, thirty platform middleware products, sixteen Extraction Transport and Transformation tools and 34 integration brokers. Since 1994, more than forty vendors have introduced integration products that use a messaging infrastructure to move data between dissimilar systems.

Why has this explosion in integration brokers come about?

Some of the driving forces behind EAI can be readily identified:

- ◆ The Internet Revolution and the New Economy
- ◆ The volatile nature of the global economy – giving rise to an unprecedented number of corporate mergers, acquisitions, divestitures and reorganizations with the corresponding need for EAI
- ◆ The trend towards downsizing and distribution of the typical IT infrastructure during the Nineties which increased the heterogeneity and diversity of products, platforms and application interfaces
- ◆ The expanding availability of robust messaging backbones (like IBM's MQSeries) in the marketplace
- ◆ The trend towards higher costs of labor and IT professional expertise in general which increases the need for simple, flexible, maintainable, extensible and re-useable software components
- ◆ The mushrooming growth of three-tier architectures driven by the need for legacy application integration, the *e-Business* revolution and *e-Commerce*
- ◆ The emergence of object oriented technologies that lend themselves to modularity and encapsulation
- ◆ The growing customer expectation of zero-latency and high availability which cannot be realized with monolithic architectures

Recently there has also been a movement towards "virtual enterprises" that must link application systems from different companies in the supply chain, especially for *e*-Commerce.

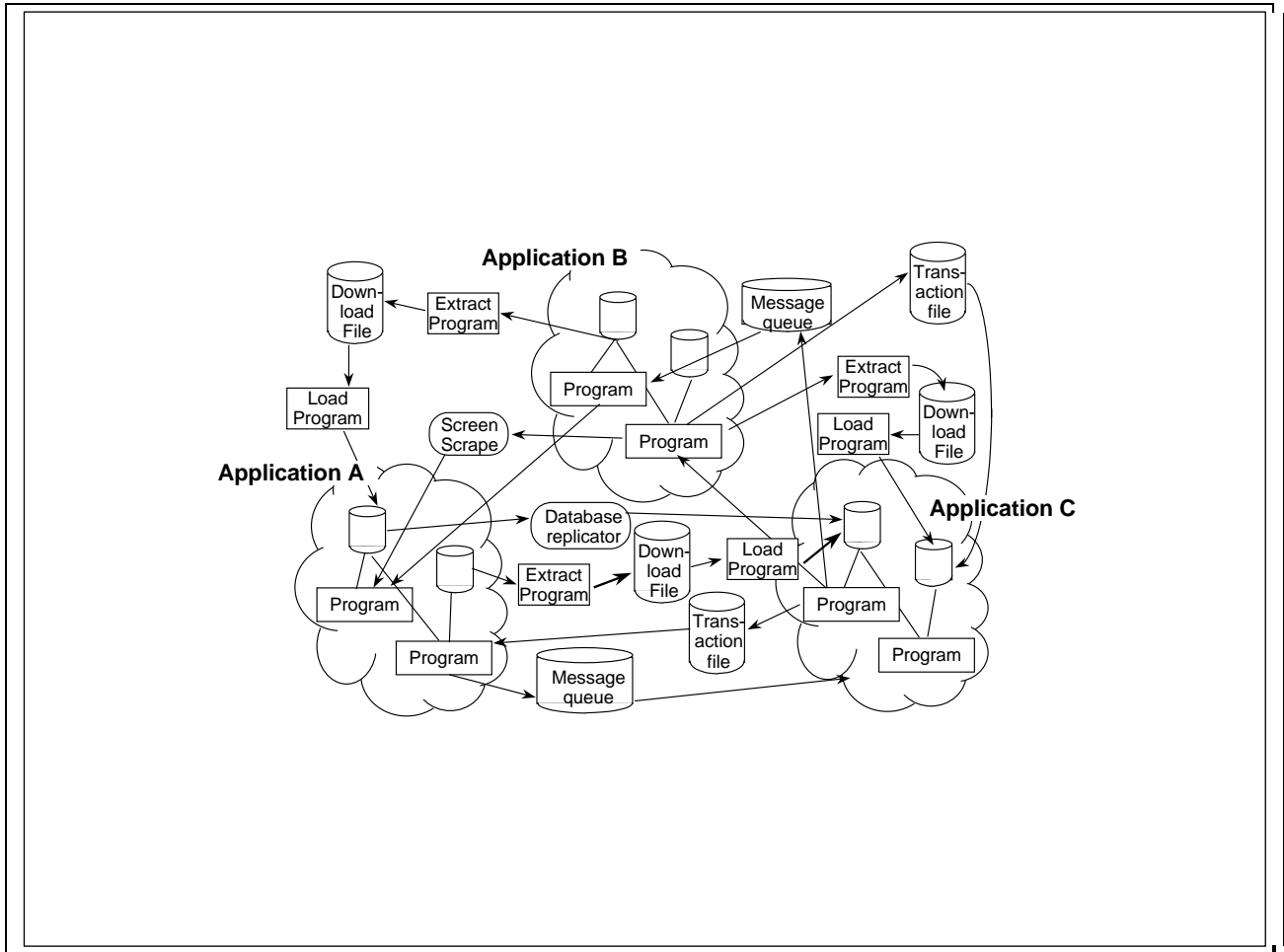
Standards will not resolve most semantic transformation complications in today's EAI market. EDI provides a good case study to illustrate this point. After more than 20 years of wide acceptance and adoption, EDI implementation still requires organizations to negotiate message semantics.

There are several reasons for this:

- ◆ Developers of packaged software use different semantics in an effort to differentiate their products
- ◆ Even in the same industry, enterprises create unique business processes and then define their data in ways that reinforce that uniqueness
- ◆ Standards are continually evolving to compete with other technologies which creates a versioning problem – semantics of new applications do not match the semantics of legacy systems
- ◆ Standards are seldom detailed enough to provide complete application or vendor neutrality

## Integration Brokers

Integrating two independently developed applications is a challenge, not helped by the lack of application interface standards (or more usually the plethora of "standards"). GartnerGroup estimates that, as much as 30 percent of the costs associated with implementing a major packaged application will be consumed by the development of point-to-point application interfaces. An even higher percentage is spent on ongoing maintenance of these interfaces. This is the primary justification of integration brokers for EAI.



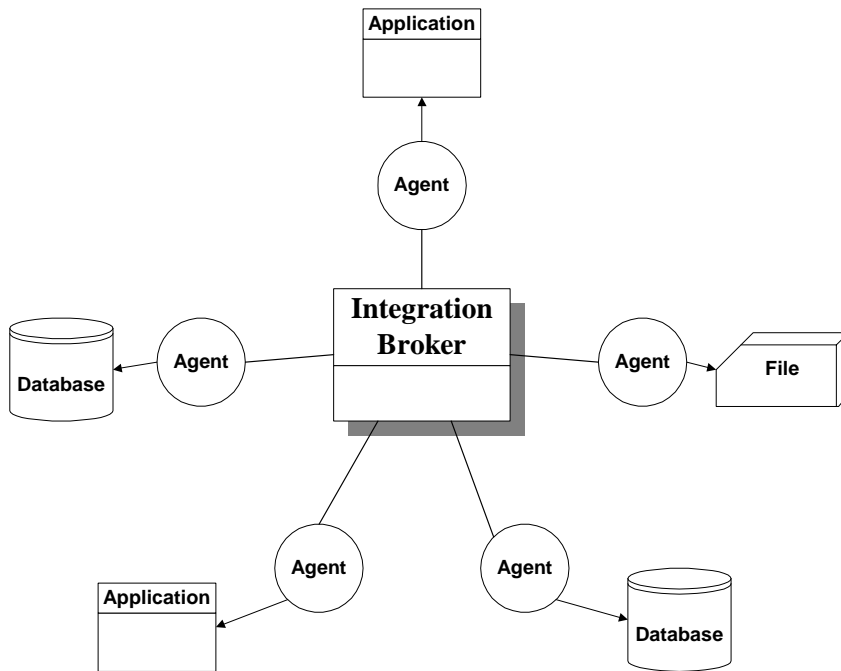
**Figure 1. Point-to-point Application Spaghetti**

The demand for Integration Brokers has grown out of the following:

- ◆ The need to reduce “inter-application spaghetti”
- ◆ The requirement for centralized control, monitoring, administration and maintenance
- ◆ The demand for simplicity (and reduction in cost) in development, deployment, and maintenance of an enterprise IT infrastructure
- ◆ The need for zero-latency – failover, load-balancing, and intelligent routing/workflow management
- ◆ The need for Business Process Automation
- ◆ The need for a unified user interface – IDEs, monitors and management interfaces

To approximate zero latency, to maximize use of best-of-breed applications, to integrate new acquisitions and to develop effective *e-Commerce* functionality, enterprises need to integrate applications that were built at different times, by different groups, on different platforms, using different technologies to communicate. To solve this problem, an integration solution should target these criteria: one connection point per application, low cost to add applications to the solution, low cost to upgrade applications, high throughput for messaging and processing, and low latency. The GartnerGroup perspective is that the most effective approach to application integration involves the use of an Integration Broker.

## The Case for “Broker-less” or Modular Integration



**Figure 2. Traditional Broker Architecture**

Almost all integration solutions offered in the market today have centralized broker architectures. This is far from an ideal situation because a brokered architecture suffers from the following pitfalls:

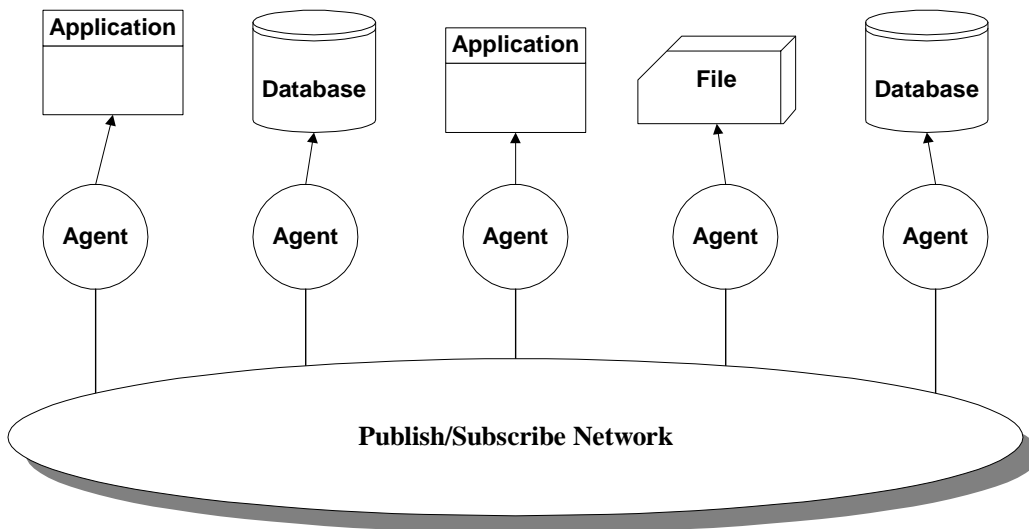
- ◆ The broker tries to be “everything to everyone” – rules-engine, data transformation-engine, translation engine, routing-broker, messaging middleware and integration engine – in one monolithic application. All too often it ends up only offering a less-than commercial-strength bundle of proprietary “bolt-ons”
- ◆ A broker architecture creates a single point-of-failure and a performance bottle-neck that is hard to avoid (even with brokering federation schemes)
- ◆ The broker is a proprietary “black-box” that ensnares you in a vendor lock-in situation
- ◆ Often, the broker becomes an unmanageable morass of ever growing business rules and undocumented procedures
- ◆ The simplistic graphical wiring tools that many vendors offer with their brokers, while superficially impressive, lead you down a path of low object-reuse, an investment in indecipherable and un-maintainable business logic, and difficult or impossible error analysis
- ◆ The broker creates longer transaction paths, because messages often need to be referred to the broker for servicing and routing (sometimes unnecessarily)
- ◆ Brokers seldom offer the facilities to allow you to migrate them to other environments, so you have trouble managing separate development, test, and production regions
- ◆ B2B integration is made more difficult if your business partner does not have the same integration product. Even with compatible products the “ownership” of a centralized broker can become a hotly contested issue

A new generation of integration solutions is now emerging to address these deficiencies. These “Second Generation” EAI solutions are characterized by modular, component-based architectures. In general,

distributed nodes in the integration network communicate peer-to-peer and use a publish/subscribe event model to circumvent the problems inherent in the old point-to-point and broker-centric methodologies.

The new breed of products is much closer to the ideal of “plug-and-play” or turnkey software. They generally embrace a view that integration components are simply “pluggable” modules that provide dedicated services and functions in a distributed integration network. The user is free to select best-of-breed solutions from multiple vendors in order to complete the integration jigsaw – rules engines, translators, workflow products and messaging middleware – as needed.

Therefore, it is crucial to the newer class of software to adopt ubiquitous Open Standards, like Sun’s J2EE technology, so that modules can interoperate.



**Figure 3. Modular Integration Architecture**

BeePlex is just such a “Second Generation” EAI solution that capitalizes on the successes and failures of the past integration initiatives.

BeePlex is process-driven, rather than data driven. Business events are broadcast on an enterprise-wide Publication/Subscription Network. Event Publishers are entirely de-coupled from event subscribers – providing an inherently flexible, asynchronous system. Subscribing components register their interest in events based on topic strings or information streams (which might include wildcards). This enables you to stream your data and create clean, organized, object-oriented integration networks.

Business events are published in Extensible Markup Language (XML), reducing message versioning, translation, portability and transformation issues to a minimum.

BeePlex operates with a network of distributed agents that communicate using the Java Messaging Specification (JMS). One or more centralized graphical consoles control the agents via TCP/IP. Each agent has one or more customizable Adapters or Sub-tasks that run as system processes and can be configured, started, stopped or disabled from a remote console. A large selection of stock adapters is provided for specific functions or third-party application interfaces.

In the past, many integration vendors have vainly tried to convince the market that integration can be achieved without resorting to application programming. Like the empty promises of the latter-day “4<sup>th</sup> Generation Language” vendors, who unsuccessfully tried to convince the IT industry in the late Eighties that coding was a thing of the past, the modern market is starting to see once again that there is no EAI

“silver bullet”. At SolutionsWorks we subscribe to the view that application programming is inevitable in all but the most trivial of integrations. If this position is accepted as fact, then the question becomes: “What is the best way to tackle integration application programming?”

Java has unquestionably established itself as the industry programming language of choice. Its robustness, reliability, security, and platform neutrality have made it the ubiquitous programming language for new application development. BeePlex is a pure Java solution that embraces the J2EE specification. Your customization and extension of the product, including the implementation of your business logic, is all in standard Java. This gives you a wide skills base to staff your integration projects from; it promotes cost reduction since the code you invest in is re-useable, and it allows you to take advantage of the substantial amount of emerging add-on products that comply with the Java Standard.

The primary features of BeePlex are:

- ◆ It eliminates the complexity of Point-to-point Integration
- ◆ Offers mature, usable, tools and interfaces
- ◆ Customization and system extensions are in standard Java – without proprietary scripting languages or graphical wiring tools
- ◆ Allows a seamless bridge between heterogeneous messaging networks
- ◆ Its modular architecture allows you to select best-of-breed components rather than proprietary, single-vendor offerings
- ◆ It eliminates single points-of-failure and system bottle-necks
- ◆ BeePlex embraces Open Standards
- ◆ Allows rapid deployment with a moderate learning curve

## Selecting an Integration Broker That’s Right for You

### *Alternatives to BeePlex - Developing your own Integration Broker*

To make the case for building a homegrown integration broker you need to consider the following:

- ◆ The costs and benefits
- ◆ Platform selection – price, performance, availability, in-house expertise
- ◆ Component selection – transport, transformation, routing/workflow
- ◆ Identify and build adapters
- ◆ Designing and developing routing/workflow managers
- ◆ Developing an IDE
- ◆ Deployment tasks
- ◆ Maintenance tasks
- ◆ Management and monitoring
- ◆ Security

Once the integration project has been sized, you are in a position to calculate a cost-benefit analysis based on using an Integration Broker like BeePlex versus the cost of development without one.

Estimated cost of first year development without an Integration Broker	\$1,000,000
Actual cost of first year development with an Integration Broker	\$700,000
First Year Savings	\$300,000
Cost of Integration Broker Implementation	\$250,000
Savings as a percentage of Implementation Costs	120%

**Figure 4. Hypothetical Financial Benefit Calculation for First Year Integration Broker Use**

Since most of the benefits that are generated by the use of an Integration Broker result from the architecture, they are derived regardless of the specific tools used to implement the Integration Broker. Because it is difficult to build a functionally representative broker from scratch, it is unlikely that the benefits accrued from in-house development will be cost effective when compared to products immediately available in the marketplace.

Ross Altman (Research Director, GartnerGroup Research and Advisory Services) says: “It is difficult for enterprises to build and then deploy a homegrown integration broker architecture, and few homegrown toolsets will deliver the functionality of the current crop of commercial products. In addition, once deployed, a ‘not invented here’ syndrome makes it difficult to get all the enterprise’s development teams to use a home-grown system, reducing the payback even further.”

### ***Selecting a Packaged Integration Broker***

Ranking integration brokers can be done based on the following criteria:

1. Price
2. Reliability
3. Platform Coverage
4. Performance
5. Range of Adapters
6. Transport mechanisms
7. Deployment Management/Monitoring features
8. Message Management/Monitoring features
9. Ease of Use/Development/Customization
10. Bulk data capabilities

Currently, only one EAI offering stands out significantly from the rest according to these evaluation criteria – BeePlex.

### ***Choosing BeePlex as Your Integration Solution***

What does BeePlex have to offer?

- ◆ Faster completion time for *e*-business integration projects (enterprise integration and B2B integration) because BeePlex comes as close to a turn-key solution as possible in the complex world of business integration
- ◆ A solid integration framework to catapult future integration projects into over-drive because you reap the benefit of having an extensible, re-usable, and flexible architecture
- ◆ A lower total investment because custom code is reduced to the absolute minimum, your existing systems are aligned without the need for invasive changes or the requirement to break open legacy code
- ◆ With BeePlex's Open Architecture (particularly the use of J2EE technology and the Java Standard) you are free from vendor lock-in and you enjoy the benefit of selecting specialized OEM components to round out your integration solution
- ◆ BeePlex exposes you to lower risk – its inherently modular architecture means that you can take advantage of a phased roll-out, and you don't need to commit yourself to technologies or methodologies that your organization might not be ready for
- ◆ A publish/subscribe model of business event handling means that your integration solution stays organized – unlike point-to-point or stove-pipe integrations that over time lend themselves to spaghetti architectures
- ◆ Your application programmers do not need to learn new APIs, scripting or macro-languages, or worry about auditing, reporting, error handling or error-correction – BeePlex frees them up to concentrate on business logic

Messaging middleware is the most common transport mechanism for an Integration Broker, and should be considered an essential prerequisite in the decision to select a broker product because of the flexibility, reliability, extensibility and prevalence of messaging middleware products. Most Integration Brokers rely on or support IBM's MQSeries (WebSphere MQ) for message transport services. Although BeePlex is vendor neutral (supporting any JMS provider messaging middleware implementation), we recommend MQSeries for your message transport layer.

(See: [www.ibm.com/software/ts/mqseries](http://www.ibm.com/software/ts/mqseries) )

## Why the BeePlex Solution Makes Sense

- ◆ **Flexibility** – the solution is cheaply and easily customized, re-organized, or even removed
- ◆ **Modularity** – a successful solution cannot be monolithic, therefore BeePlex has replaceable, discrete components that plug-in, interact and intercommunicate
- ◆ **Asynchronosity** – your integration solution must not depend on the immediate availability of distributed components (brokers, remote servers etc.) in order to function. This is achieved with the use of a robust messaging system that has two-phase commit and store and forward messaging capabilities
- ◆ **Extensibility** – BeePlex has a natural, well-defined growth path. As the use of the system grows, and the number of business tasks it performs multiplies and the internal complexity of the system increases. BeePlex is able to absorb new requirements with the minimum of impact on the existing infrastructure
- ◆ **Scalability** – as the solution is extended (or the volume of usage increases) the impact to the system is linear in fashion, without bottlenecks or exponential use of resources
- ◆ **Reliability** – BeePlex is robust. It has no single points-of-failure and takes advantage of Java's reliability
- ◆ **Performance** – BeePlex promotes the highest data velocity possible, with the minimum of CPU, memory, storage, and network overhead. The system as a whole tends towards zero-latency.

- ◆ **Maintainability** – our use of pure Java means that skills to enhance your integration are freely available in the industry
- ◆ **Simplicity** – we offer a solution that is uncomplicated and elegant, without compromising the features listed above

With the open architecture and commitment to Open Standards that BeePlex offers, your investment has a future.

## About SolutionsWorks

SolutionsWorks is a privately funded Oregonian company. It was founded in May 2005 with the vision of creating reliable, robust, and - most importantly - usable technology. We like to say that, in a word, we strive to create *elegant* solutions.

We aim to apply technology to niche areas that have not traditionally been associated with high-tech in the past. SolutionsWorks is uniquely positioned to assimilate the full range of new low power and inexpensive wireless ZigBee networks into the enterprise computing environment.

For more information please email [info@solutionsworks.com](mailto:info@solutionsworks.com)

Copyright © 2008 SolutionsWorks Inc.

Corporate Headquarters:  
SolutionsWorks Inc.  
2828 SW Corbett Ave. #117  
Portland, OR 97201  
503-343-9984  
Fax 415-358-5889